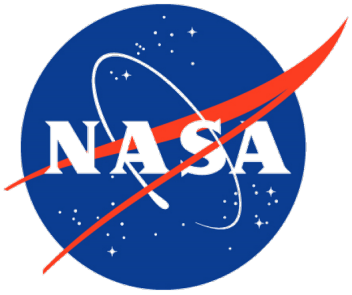

Bachelor's Thesis

Space Mission Scheduling Toolkit for Long-Term Deep Space Network Loading Analyses and Strategic Planning

by

Guillem Rueda Oller

July 2019

Degrees
Bachelor's Degree in Aerospace Technology Engineering
Bachelor's Degree in Informatics Engineering

Supervisors
Kar-Ming Cheung
Marc Sánchez Net

Tutor
Antoni Grau Saldes

This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

Space Mission Scheduling Toolkit for Long-Term Deep Space Network Loading Analyses and Strategic Planning

by

Guillem Rueda Oller

Abstract

The Jet Propulsion Laboratory (JPL) owns and operates the Deep Space Network (DSN), a set of antennas placed around Earth to communicate with spacecraft flying anywhere in the Solar System. While the DSN is a critical asset to JPL and NASA's success, it is also expensive to build, maintain and operate. Therefore, additional system capabilities are planned strategically, years in advance, by forecasting which missions will utilize the system in the coming decades (and their driving data requirements). Then, loading analyses are conducted assuming different scenarios, each one simulating DSN operations for several years.

Within this context, this thesis focuses on developing an automated long-term scheduling mechanism that can mimic real DSN operations. Several factors are modeled and accounted for in this process: Spacecraft visibility constraints, evolution of the DSN architecture, characteristics of each antenna, as well as link and other operational constraints.

To implement the scheduling mechanisms, several options are first identified and downselected. Then, it is explained in detail how the automated long-term scheduling toolkit –LTST– formulates the problem as a mixed integer linear programming (MILP) problem. The procedure to obtain the constraints of the optimization problem is presented and the objective function is defined.

Finally, results of several case inputs are presented, including very oversubscribed scenarios. Computational performance of the toolkit is evaluated using real inputs from the DSN system engineering team and we show that the thousands of tracks for tens of missions can be successfully scheduled within reasonable computational complexity.

Keywords: Deep Space Network, space communications, scheduling, mixed integer linear programming, loading analysis

Thesis Supervisor: Kar-Ming Cheung
Title: Technical Group Supervisor at NASA Jet Propulsion Laboratory

Thesis Supervisor: Marc Sánchez Net
Title: Telecommunications Engineer at NASA Jet Propulsion Laboratory

Acknowledgments

First, I would like to truly thank Dr. Kar-Ming Cheung and Dr. Marc Sánchez Net for their invaluable guidance and support provided during my stay at JPL. With your excellent advice and supervision I managed to carry out this project. It has been a pleasure to work with you.

Many thanks to all the people in section 332, specially the ones on floor 4 in building 238. You made my day for six months. Also, I am very grateful for all the wonderful hours spent with other interns I met throughout my stay in Pasadena.

I would like to specially thank the CFIS program. First, I am very grateful to them for coordinating my double-degree program. Second, this extraordinary opportunity, half year doing research at NASA Jet Propulsion Laboratory, would not have been possible without the intervention of the CFIS. Third, thank you for providing financial help throughout my whole studies at UPC. Special thanks to Fundació Privada Cellex for funding my stay at JPL.

After five unforgettable years at UPC, this thesis marks the end of my undergraduates studies in Spain. I am extremely grateful to all the amazing people I have met these years. We have had uncountable memorable moments. Thank you.

Last but not least, I want to thank the tremendous support received from my family and friends during this time. Big thanks to my parents and my brother for always encouraging me to follow my dreams.

Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

1. Introduction

1.1 Context

All space missions need to send scientific data to Earth. In fact, data produced by a spacecraft is only useful if it is processed and analyzed by the scientific community on Earth. Most deep space missions never return to Earth, so a communication system is essential to send data to Earth and receive commands and data from Earth. Indeed, the spacecraft's tracking and communications system is the only way to communicate and interact with the mission.

NASA uses the Deep Space Network (DSN) to support interplanetary spacecraft missions, selected Earth-orbiting missions, and to perform radio and radar astronomy observations for space exploration [3]. The DSN is the ground segment of NASA's communications system for deep space missions. It currently consists of three antenna complexes at different locations situated approximately 120 longitude degrees apart on Earth [4]. This architecture ensures continuous coverage and tracking for deep space missions. Today, each complex has one 70-meter antenna and a few 34-meter antennas [4]. They can be used individually or in combination to satisfy the communications requirements of each mission [4].

The demands placed on the DSN are continuously increasing. For example, the Mars Reconnaissance Orbiter (MRO) sent to Earth more than 298 terabits of data by March 2016 [4]. However, NASA estimates that the deep space communications capability will need to grow by nearly a factor of 10 each of the next three decades [4]. Moreover, the ability to answer an increasing number of detailed scientific questions requires more sophisticated instruments that generate even more data that later will be transmitted to Earth.

Unlike the Internet, where protocols require millisecond-level latencies, deep space communications have latencies or disruptions of up to several hours because of the enormous distances to which spacecrafts travel [4]. For example, the roundtrip light time to the Voyager spacecraft and back is more than 24 hours [4]. Because of this, and adding the fact that missions need to send a huge amount of data (hours), scheduling is necessary in the DSN.

While the DSN is a critical asset to JPL and NASA's success, it is also expensive to build, maintain and operate. Therefore, additional system capabilities are planned strategically, years in advance, by forecasting which missions will utilize the system in the coming decades (and their driving data requirements). Then, loading analyses are conducted assuming different scenarios, each one simulating DSN operations for several years. Within this context, this thesis focuses on developing an automated long-term scheduling mechanism that can mimic real DSN operations.

1.2 Motivation

NASA spends significant resources in the Deep Space Network (e.g., approximately $23 was spent for the construction of the 34-meter antenna DSS-35, in Canberra [5]). For example, resources are spent building new antennas, maintaining current stations or upgrading obsolete equipment of the network. Although all these procedures are extremely expensive, they are necessary to support NASA's current and future missions. Therefore, it is necessary to forecast communication needs of future missions and plan for DSN upgrades strategically, years in advance.

Moreover, the DSN is and will be a highly utilized asset. Indeed, nowadays not all requested communications in the DSN are finally executed. That happens when the network is oversubscribed, such as in special events. In order to correctly decide which new DSN capabilities should be implemented, loading analyses of different long-term scenarios of the DSN are necessary. This motivates the development of a long-term DSN scheduling toolkit, which is the main goal of this thesis.

1.2.1 State-of-the-art in DSN scheduling

Scheduling the Deep Space Network has been addressed since the very beginning of the network. It can be classified in short-term, mid-term and long-term scheduling. Several DSN scheduling toolkits, each with its strengths and weaknesses, have already been implemented. This thesis addresses the long-term case. However, since long-term scheduling has several common features with short-term and mid-term DSN scheduling, the state-of-the-art of all time ranges is presented. In particular, this section presents three projects where a DSN scheduling toolkit was successfully implemented. Furthermore, for each project we justify why the new scheduling toolkit developed in this thesis is needed.

Service Scheduling Software

The foremost work in DSN scheduling is led by Mark D. Johnston at Jet Propulsion Laboratory (JPL). The creation of the Service Scheduling Software (S^3) was culminated in 2011. It only focused on the mid-term process – from 2 weeks to about 6 months before execution. The team implemented S^3 as a web application with a user interface [6]. According to Johnston, due to the collaborative peer-to-peer negotiation nature of the mid-term DSN scheduling [7, 8], S^3 was heavily driven by the need for users from multiple time zones to participate in distributed conflict resolution and change negotiation [6]. In 2016, S^3 was extended to support real-time DSN scheduling, which enabled consistent data model to be used for mid-term, short-term and real-time scheduling [6].

S^3 uses the DSN Scheduling Engine (DSE). The DSE expands scheduling requests into individual communications tracks, identifies potential conflicts in the schedule and conflict-free allocations, and then attempts to find a suitable solution for each. The DSE is based on running multiple instances of ASPEN (Automated Scheduling and Planning ENvironment), the planning and scheduling framework previously developed

at JPL [9]. The DSE is based on three algorithms: First, an initial layout algorithm is executed to initially generate tracks that satisfy a request's specifications. This first algorithm goes request per request systematically searching for legal intervals to satisfy it. Second, a basic repair algorithm is executed to reduce conflicts and violations. It tries to solve each conflict or violation in the schedule until all are solved or a timeout expires. Finally, an algorithm that tries to extend the scheduled tracks to the their minimum preferred length is applied [9].

The drawbacks of S^3 that justify the new scheduling toolkit developed in this thesis are explained below along with Johnston's Loading Analysis and Planning Software.

Loading Analysis and Planning Software

In 2018, Johnston added support for long-term planning and forecasting. This led to a newly completed component called LAPS – Loading Analysis and Planning Software, which extends the S^3 framework, so that they both share the same data model [6]. A forecast run in LAPS goes through the following four computational steps: First, for each mission planning request, constraining time range, potential assets or asset sets, and visibility intervals are determined. Second, the distributed request track duration is accumulated over all the potential asset options. Third, weights are adjusted to lower oversubscription on each asset. Fourth, oversubscription is tabulated by asset group and by mission, for aggregation and reporting [6]. In conclusion, the main idea of the algorithm is the same as in S^3 – initial layout and then repair.

The main drawback of S^3 and LAPS is in the scheduling algorithm. The algorithm in both software is clearly based on a track-by-track sequence. This means that for each new track request, the algorithms looks for a legal interval for that track, ending this stage with an initial schedule. Then, it tries to repair each conflict or violation until a timeout is reached. Therefore, there is no guarantee that these tools can provide the optimal schedule, especially in the case of an oversubscribed resource like the DSN (a very common situation in real-life operations[1]). Actually, the fact that it tries to schedule each track when creating the initial layout and then repair the schedule, does not always give an optimal schedule, because the result is very dependent of the order of the input data. In contrast, the main research goal of this thesis is to implement a DSN scheduling toolkit that achieves the best possible solution (or a very good one) in long-term oversubscribed scenarios, globally optimizing the schedule. Instead, Johnston's work focuses more on providing a unified user-friendly toolkit, even letting the user choose how to solve some conflicts in S^3 [6], rather than focusing on the quality of the solutions, specially in highly oversubscribed intervals.

Link-Capability Driven Network Planning and Operation

Research by Cheung, Lee, Gearhart, Vo and Sindi (2002) presents a network planning and scheduling concept that takes into account communication link capabilities and telecom performances to improve network communications efficiency. The key idea of the work is that the resulting schedule will be operating at favorable telecom config-

[1] Learned at Jet Propulsion Laboratory (2019)

urations, which means higher data rates and therefore time allocated to each mission can be shortened, increasing the number of missions that the network resources can support [10]. Cheung et al. (2002) present a mathematical framework consisting of a nonlinear constrained optimization problem, where the sum of the time allowed for all tracks is minimized.

The formulation as a optimization problem is of great interest for this thesis, as the problem is globally optimized leading to the best solution. However, there are three disadvantages in its formulation: (1) All tracks in the present in the optimization problem must be scheduled; (2) It seems that each track has predefined which ground station it will use when formulating the mathematical problem; and (3) it does not take into account prioritization between tracks or other characteristics, thus just focusing on increasing the total data volume [10]. Disadvantages (1) and (2) arise from the fact that no if-then conditions are defined in the mathematical problem presented. Also, (3) happens because all tracks are finally scheduled, so no prioritization is needed. Moreover, the problem has a nonlinear constraint. A sample communications network consisting of three tracking ground stations and six orbiting satellites is simulated and optimized. In the test run they assume that each track must constitute at least 20 minutes in length, so there are exactly 31 passes [10].

The work presented is clearly not for long-term scheduling, but short-term scheduling, where the schedule is never oversubscribed. Moreover, assignments between ground stations and spacecrafts are not included in the optimization problem. Still, Cheung et al.'s (2002) concept of global optimization may be useful in some section of this thesis.

Architecture Loading Analysis Tool

In Reference [11], MacNeal et al. (2016) discuss the results of preliminary loading simulations for hybrid RF-optical network architectures and highlights key mission and ground infrastructure considerations that emerge. They introduce the Architecture Loading Analysis Tool (ALAT). ALAT enables the DSN to analyze how well the projected future mission set loads up on alternative architectures consisting of both RF and Optical apertures for 20-30 years of operations [11]. Therefore, ALAT is a long-term DSN scheduling tool which also incorporates optical antennas, despite the fact that the DSN currently has no optical communication capabilities yet.

The scheduling algorithm used in ALAT is a greedy in nature – it makes the optimal choice at each step. When the algorithm has a new track to schedule, it attempts to schedule it at the complex with the longest available visible time segment. It then tries to schedule it using antennas at that complex that still have enough available time and can support the spacecraft. If that is not possible, it then attempts to schedule a partial track at the complex, or reject it [11].

The main point to highlight here is that once a track is scheduled, it cannot be rescheduled to allow new tracks to be successfully scheduled. This work, similar to S^3 and LAPS presented before, is based in a greedy algorithm. Consequently, ALAT does not find a global optimal solution to the DSN scheduling problem. Moreover,

the greedy algorithm in ALAT can result in largely sub-optimal schedules in oversubscribed scenarios, because at a certain execution point, it will reject all new tracks. Therefore, the result is highly dependent on the order of the input data. However, this tool will probably influence the toolkit developed in this thesis, as the new toolkit is intended to replace ALAT.

Other DSN scheduling toolkits

Prior to the tools already described, several authors had addressed the problem of scheduling DSN resources. For instance, LR-26 was a customizable heuristic scheduling system for the 26-meter antennas subnetwork that has now been decommissioned. It used Lagrangian relaxation and constraint satisfaction search techniques [12]. The Operation Mission Planner (OMP-26) used heuristic search to allocate 26-meter antennas to missions, and linear programming to adjust track durations [13]. DANS, which stands for Demand Access Network Scheduler, included all antennas and used a heuristic iterative repair approach [14]. These and some other investigations are now obsolete because they were focused on past DSN configurations. Moreover, these software were replaced by the toolkits detailed above.

Finally, in Reference [15], Johnston (2006) presents a multi-objective approach to the scheduling problem of the Deep Space Array-Based Network (DSAN). The paper uses evolutionary algorithms to solve the problem. This approach may be good for this thesis, but this will be discussed in a further section. In any case, the software presented is implemented exclusively for DSAN, a network architecture that is now not considered feasible for future DSN enhancements.

1.3 Background

1.3.1 The Deep Space Network

The **NASA Deep Space Network (DSN)** is a worldwide network of spacecraft communication facilities managed and operated by NASA Jet Propulsion Laboratory (JPL). The DSN supports interplanetary spacecraft missions, radio and radar astronomy observations to explore the Solar System, and support selected Earth-orbiting missions [3]. The DSN is operated at all times during the year [16]. This section provides a brief summary of the key concepts and characteristics of the DSN and deep space communications necessary to understand the rest of the work.

DSN Architecture

The DSN is the largest and most sensitive scientific telecommunications system in the world today. It consists of three **Deep Space Communications Complexes** (DSCCs), placed approximately 120 degrees apart around the world: at Goldstone, near Barstow, in California's Mojave Desert; at Robledo near Madrid, Spain; and at Tidbinbilla near Canberra, Australia. The **Network Operations Control Center** (NOCC) is in building 230 at JPL. The strategic placement of the DSCCs allows uninterrupted tracking of any interplanetary spacecraft as the Earth rotates [3].

The International Telecommunications Union, which sets aside various frequency bands for deep space and near Earth use, defines that an spacecraft is in "deep space" when it is further than 2 million kilometers from the Earth's surface [17]. The DSN was designed to communicate with spacecraft traveling approximately 16,000 km (10,000 miles) from Earth to the farthest planets of the Solar System [18]. Figure 1-1 shows that spacecrafts at an altitude of 30,000 kilometers or more are always in the field of view of a DSCC [1].

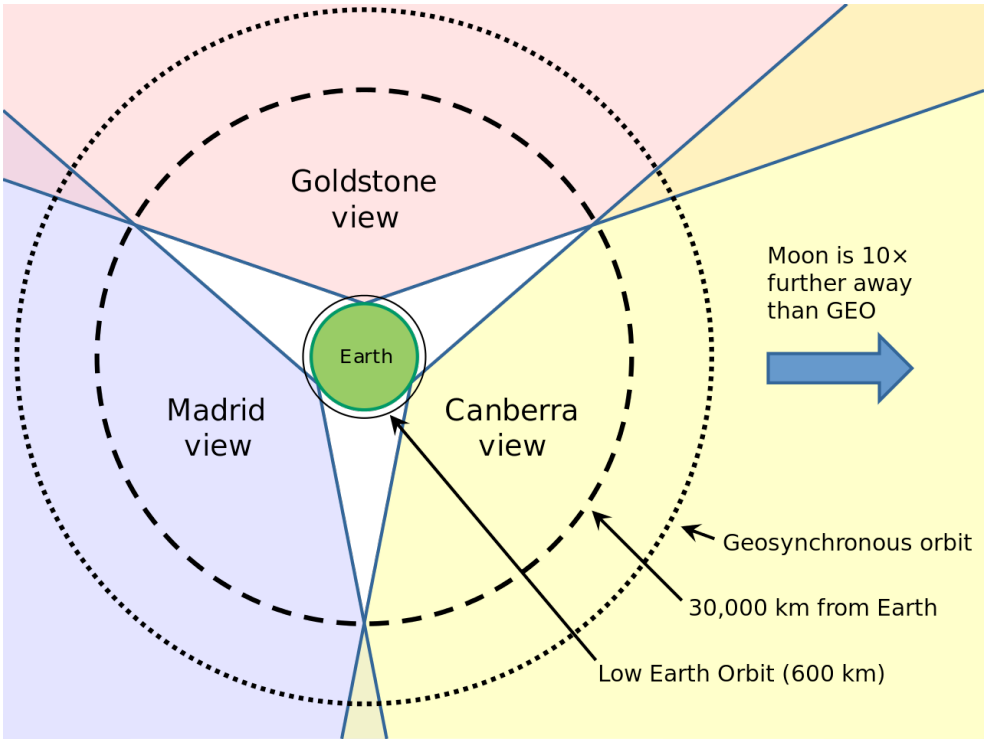

Figure 1-1: Field of view of the Deep Space Network [1]

All three DSCCs have generally the same makeup, although Goldstone (GDSCC), being closest to JPL, has some additional antennas, as well as research and development facilities not found at Madrid (MDSCC) or Canberra (CDSCC). Each DSCC has a number of **Deep Space Stations** (DSSs). Each DSS comprises a high-gain steerable parabolic-reflector (or antenna dish), and its associated front-end equipment such as low-noise amplifiers and transmitters. Throughout this thesis, the term antenna will refer to the whole DSS. Also, each DSCC has one **Signal Processing Center** (SPC), which connects with all the DSSs at the DSCC, and houses the operations personnel along with the computers and other equipment [3].

Today, each complex has one 70-meter antenna and a few 34-meter antennas [4]. Each antenna supports frequencies in S-band, X-band, K-band and/or Ka-band [17]. The Deep Space Network also provides back up to other two networks: the Near Earth Network (NEN) and the Space Network (SN) [16].

Figure 1-2: DSS-43 (70-meter antenna) in the foreground with 34-meter antennas in the background in Canberra [2]

DSN Now [19] is a web application that presents information on NASA's Deep Space Network. The application provides, in a highly visual format, data on spacecraft that the DSN stations are communicating with at any given time.

DSN Operations

Consider a space mission as any spacecraft, satellite, missile, rover or space station outside Earth. Within this thesis, the term ***deep space mission*** will be used to refer to any mission in deep space as well as any mission in selected Earth orbits (i.e., any mission supported by the DSN).

Each deep space mission is typically broken down in multiple operational segments. An **operational segment** is a period of time where the communication needs of a mission (and thus the DSN support) are approximately constant (e.g., same contact time, link budget, contact periodicity). An operational segment has a start date and an end date as well as a type (e.g., DDOR, telemetry, science, etc.). Usually only one operational segment per mission is active at a time. However, two or more operational segments of a mission may be occasionally active at the same time, specially when each operational segment is of different type.

A **track** or pass is a communication opportunity between a space mission and the DSN. Each track has a specific length, technical requirements (link budget) and a time window to be scheduled. A track may establish more than one link simultaneously.

A **link** is physical communication channel established between a spacecraft and a DSN antenna for the purpose of communicating data and/or acquiring tracking information. Three types of links are considered in this thesis:

- **Uplink:** Only information from Earth is sent to the mission. The Equivalent Isotropically Radiated Power (EIRP) of the DSN antennas used is the main mission requirement.
- **Downlink:** Only information from the mission is sent to Earth. The Antenna Gain-to-System Noise Temperature Ratio (G/T) of the DSN antennas used is the main mission requirement.
- **Uplink&Downlink:** Information is exchanged between DSN facilities on Earth and the mission. Both EIRP and G/T of the DSN antennas used are the main mission requirements.

The uplink is used to send commands to a mission and the downlink is used to receive telemetry and science data from a mission. Additionally, these links can provide tracking of the mission too. Finally, the DSN hardware usually needs a setup and a tear-down time before and after each track respectively.

The DSN has four special track types that merit further explanation: Antenna Arraying, Delta-Differential One-Way Ranging, Multiple Spacecraft Per Antenna, and Multiple Uplink Per Antenna. These four features characteristic of the DSN are briefly explained below.

Antenna Arraying

Antenna arraying is a communication technique that combines several antennas to receive a signal. Arraying of antennas increases the G/T of the system, so signals with lower signal-to-noise ratio (SNR) can be detected. Antenna arraying typically applies to downlinks. When antenna arraying is active, an array processor is used to synchronize the signal received by each antenna forming the array. In case of an uplink&downlink, one of the antennas in the array will establish the uplink simultaneously. Currently, arraying can only be done using up to four antennas of the same ground station. Also, there is currently only one array processor in each DSN ground station, so no more than one antenna array per ground station can be active at a time.

Delta-Differential One-Way Ranging

Delta-Differential One-Way Ranging, abbreviated Delta-DOR, is a high precision tracking method for deep space missions. With regard to DSN scheduling, Delta-DOR requires establishing two downlinks simultaneously, each with a 34-meter antenna from different ground stations. Therefore, there must be full visibility from both antennas during the whole Delta-DOR track.

Multiple Spacecraft Per Antenna

Multiple Spacecraft Per Antenna or per Aperture (MSPA) allows the DSN to establish several downlinks simultaneously from different missions with the same antenna. MSPA is possible when missions are on the same region in space, such as Mars, so

that all of them are visible from the antenna's main beam. Currently, an antenna of the DSN can have up to four MSPA downlinks simultaneously.

Multiple Uplink Per Antenna

Similar to MSPA, Multiple Uplink Per Antenna or per Aperture (MUPA) allows the DSN to establish several uplinks simultaneously to different missions with the same antenna. MUPA is possible when missions are on the same region in space, such as Mars, so that all of them are visible within the antenna's main beam.

1.3.2 "The DSN Scheduling Problem"

The goal of "The DSN Scheduling Problem" (the DSN SP) is to schedule as many tracks as possible in a given time horizon, satisfying mission requirements and priorities. Although it is sometimes referred as *"DSN scheduling"* –as if it was only one problem–, the three following combinatorial problems need to be solved:

- **Assignment Problem:** Assign resources to each scheduled track. Resources assigned to each scheduled track can be either one antenna or a set of antennas, and one array processor if arraying is required.
- **Covering Problem:** When doing MSPA or MUPA, identify which resources to use for covering tracks from multiple missions at the same time.
- **Scheduling Problem:** Try to schedule each track of each mission into a DSN timeline, according to requirements and priorities.

Having to solve three different but interrelated problems largely increases the overall problem complexity. Below, each combinatorial problem applicable to DSN scheduling is discussed.

Assigning resources to tracks

In terms of DSN scheduling, the **assignment problem** individually assigns DSN resources to each scheduled track. The assigned resources are all hardware used to establish all communication links required by said track. The resources may not be shared during the same period of time with other tracks, unless in specific circumstances explained in the covering problem. The resources must satisfy the operational and technical requirements. The common requirement is that all antennas assigned to one track must be visible to the mission during all the duration of the track. The other requirement is that the link or links necessary for the track must satisfy the link budget. The link budget can be particularized depending on the type of the track, as will be explained in Chapter 2.

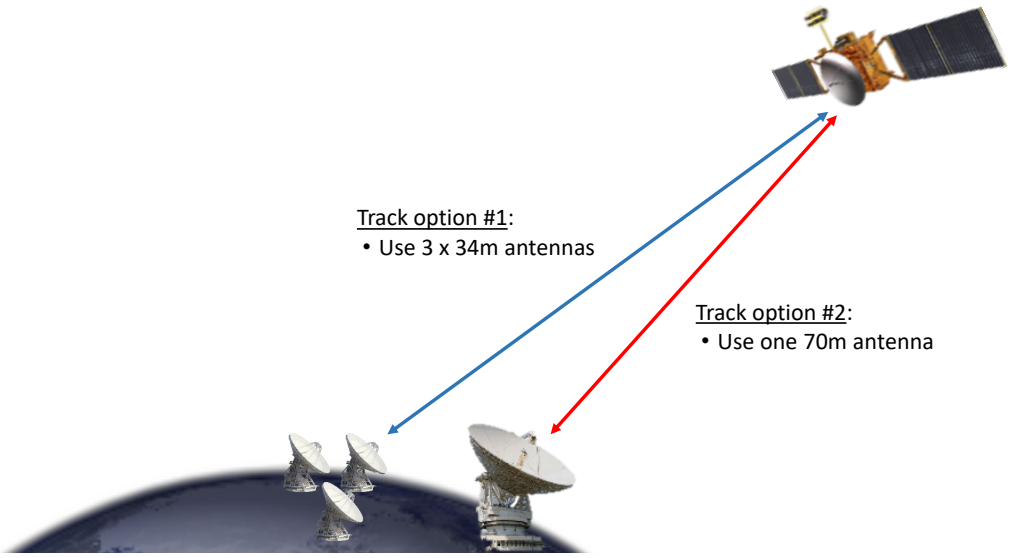

Figure 1-3: Two possible options of resources to assign to a track

The assignment problem chooses resources that optimize the problem. In other words, there can be more than one set of antennas that satisfy the link budget and visibility requirements for a track. In Figure 1-3 there is an example of assignment problem applied to DSN scheduling. The mission has two possible options that satisfy its requirements for an uplink&downlink track: (1) Use an array of three 34-meter antennas; or (2) use one 70-meter antenna. The criteria to choose the final option will depend in several factors that will be discussed in Chapter 2. In the example, a possible reasoning for choosing option 2 could be that using only the 70-meter antenna keeps the three 34-meter antennas available for up to three other missions which require only one 34-meter antenna each.

Covering missions by sharing resources

The **covering problem** –or partitioning problem– in terms of DSN scheduling consists in partitioning the set of tracks of different missions in a given moment, in subsets where each subset of tracks will use the same antenna or set of antennas at the same time. Sometimes a single antenna or set of antennas can process tracks from several missions at the same time. This can happen whenever it is possible to do Multiple Spacecraft Per Antenna (MSPA) and/or Multiple Uplink Per Antenna (MUPA), as previously explained in Section 1.3.1.

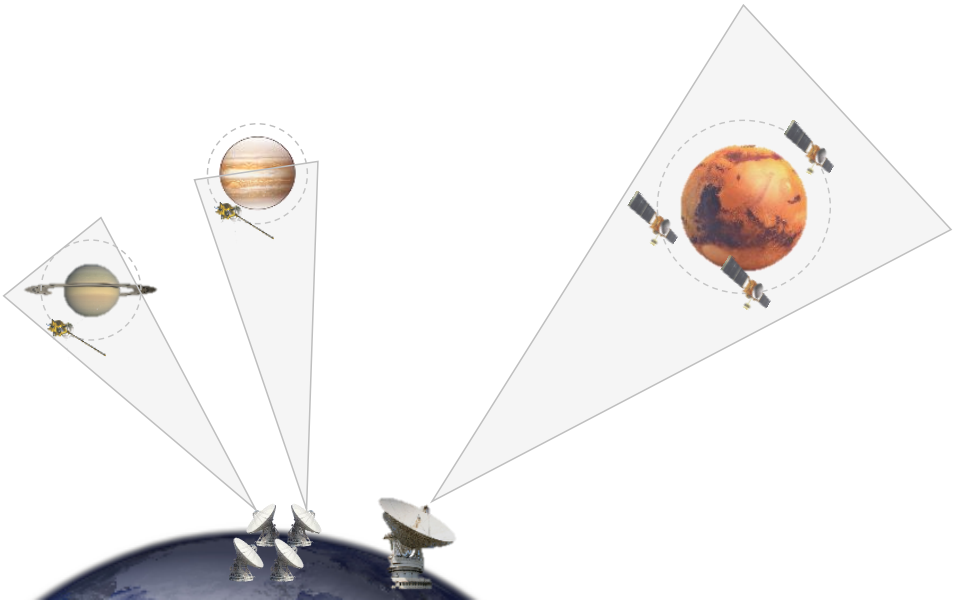

Figure 1-4: One antenna covering three missions at the same time

Figure 1-4 shows an example of the DSN covering problem. There are 5 missions that want to do a track with the DSN at the same time: one mission is orbiting Saturn, another mission is orbiting Jupiter, and three other missions are orbiting Mars. In this example, all mission tracks are grouped in three subsets, according to the planet that they are orbiting. Therefore, there is one antenna covering the three missions orbiting Mars at the same time and simultaneous links are supported using either MSPA or MUPA.

There are hypothetical cases of the DSN covering problem that do not exactly match the definition presented above. These are tracks sharing some, but not all resources with another track, or a track that shares some resources with another one and other resources with a different track, all at the same time. However, this hypothetical cases will not be considered in this research, and nor will the covering problem be solved on its own. Indeed, the input provided to the toolkit developed in this research will be the same input of the Architecture Loading Analysis Tool (ALAT). The Microsoft EXCEL®-based inputs of the DSN scheduling toolkit group each MSPA or MUPA operation as if it was one unique track [20]. Consequently, as the MSPA and MUPA operations are already specified in the input data, the DSN scheduling toolkit does not have to handle the covering problem.

Scheduling tracks in the DSN

The **scheduling problem** applied to DSN operations consists in allocating time for tracks given a finite time horizon. Subject to requirements and priorities, the scheduling algorithm schedules the requested tracks maximizing the number of scheduled tracks, the total time of DSN usage, or another metric. Is it possible that the DSN schedule is oversubscribed during some period of time, which means that there are more tracks requested than the number of tracks that the DSN can support, so not all tracks are scheduled.

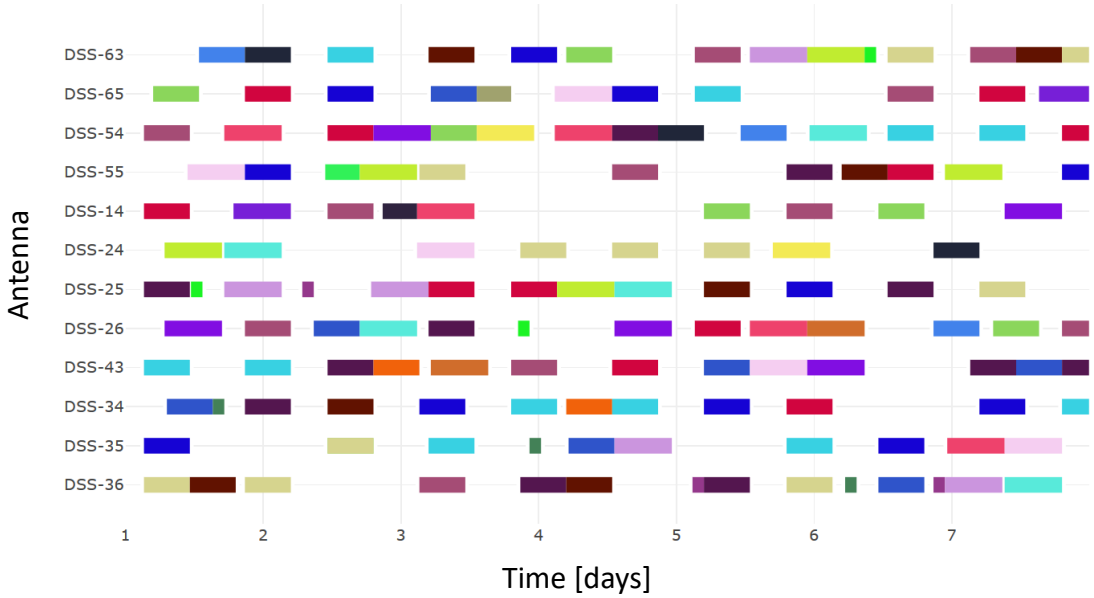

Figure 1-5: 142 tracks from 27 missions (each mission is represented by a different color) scheduled for one week

In Figure 1-5 the timeline for tracks from 27 different missions for one week is presented. It plots part of the output of a simulation using the DSN scheduling toolkit developed in this thesis. There are a total of 142 tracks with lengths of 2, 6, 8 and 10 hours. In the example, all request tracks have been scheduled in one of the 12 antennas of the DSN. Then, the schedule is not oversubscribed. How to schedule tracks in the DSN will be detailed in Chapter 2.

1.4 General Problem Statement

Section 1.3.2 provided a clear definition of track scheduling in the context of deep space communications. Furthermore, it described the problem of assigning resources of the DSN to tracks as well as covering missions by sharing resources at the same time. Combining the three problems that constitute "The DSN Scheduling Problem", leads to the formulation of the following thesis' general problem statement:

> The goal of this thesis is to develop an automated long-term scheduling toolkit to (1) conduct long term loading analyses of DSN resources under different (partially predicted) mission scenarios, and (2) inform strategic planning of future DSN investments and capabilities. This will be achieved by mapping current and future DSN resources to predicted missions tracks.

It should be clarified that, at this point of the work, the specific type of automated long-term scheduling toolkit to be developed is still not concisely defined. This will be addressed during the literature review once different options to solve the problem are analyzed. Also, the proposed general problem statement does not specify how the requested tracks are satisfied. This will be specified in the specific problem statement.

1.5 Literature Review

In order to choose how to address the DSN SP, it is necessary to do some research on the available options. In this section, several options to mathematically formulate the problem are presented. These options are evaluated and compared among them to select the one that best fulfills the needs of the thesis problem.[2]

1.5.1 Methods for "The DSN Scheduling Problem"

As explained in Section 1.3.2, the DSN SP is actually composed of three combinatorial problems: assignment problem, covering problem and scheduling problem. However, as mentioned before, there is no need to tackle the covering problem in this work because the input to the toolkit already has determined MSPA and MUPA operations. Therefore, only methods to formulate and solve the scheduling problem and the assignment problem applied to DSN operations are evaluated.

Methods for the scheduling problem

First, the scheduling problem is addressed. Pinedo (2016) presents in his book [21] how to formulate and solve different types of scheduling problems. Pinedo states that a scheduling problem is described by a triplet $\alpha \mid \beta \mid \gamma$. The α field describes the machine environment and contains just one entry. The β field provides details of processing characteristics and constraints, and may contain no entry at all, a single entry, or multiple entries. The γ field describes the objective to be minimized and often contains a single entry [21].

Based on the DSN architecture (antennas in parallel), **identical machines in parallel** (Pm) is the machine environment that must be specified in the α field. In this environment there are m identical antennas in parallel which represent m different antennas in parallel. If track j cannot be processed on just any machine, but only on any one belonging to a specific subset M_j, then the entry M_j appears in the β field [21].

The processing restrictions and constraints specified in the β field may include multiple entries [21]. For the DSN architecture, possible entries in the β field are:

- **Release dates** (r_j): Instant in time before with track j cannot be scheduled.
- **Sequence dependent setup times** (s_{jk}): Sequence dependent setup time incurred between processing of tracks j and k. If the setup time between tracks j and k depends on the antenna, then the subscript i is included (i.e., s_{ijk}). However, as in the DSN these setup times do not heavily depend on the tracks or antennas, sequence dependent setup times in the β field will not be considered. Instead, the setup and tear-down times of each track can be added to the track's original length.

[2] This section only presents the literature review of methods to solve the DSN SP. Literature review of the state-of-the-art in DSN scheduling was presented in Section 1.2.1.

- **Breakdowns** (*brkdwn*): Period when an antenna is not available. The periods that a antenna is not available are assumed to be fixed in the input data – e.g., due to shifts or scheduled maintenance.
- **Machine eligibility restrictions** (M_j): Set of antennas that can process track j. When M_j is not present, all m antennas are capable of processing track j. This entry is the result of the assignment problem.

In contrast to release dates, due dates are not specified in the β field. The type of objective function gives sufficient indication whether or not there are due dates. The due date d_j of track j represents the committed completion date. Completion of a track after its due date is allowed, but then a penalty is incurred. When a due date must be met it is referred to as a *deadline* and denoted by $\bar{d}_j$, and does go to the β field.

According to Pinedo, the objective to be minimized is always a function of the completion times of the tracks, which depend on the schedule. The completion time of track j is denoted by C_j. The objective may also be a function of the due dates [21]. The *lateness* of track j is defined as

$$L_j = C_j - d_j \tag{1.1}$$

which is positive when track j is completed late and negative when it is completed early. The *tardiness* of track j is defined as

$$T_j = \max(C_j - d_j, 0) = \max(L_j, 0) \tag{1.2}$$

The difference between the tardiness and the lateness lies in the fact that the tardiness never is negative. The *unit penalty* of track j is defined as

$$U_j = \begin{cases} 1, & \text{if } C_j > d_j \\ 0, & \text{otherwise} \end{cases} \tag{1.3}$$

The lateness, the tardiness, and the unit penalty are the three basic due date related penalty functions considered by Pinedo.

Some possible objective functions that can be minimized while solving a scheduling problem are [21]:

- **Makespan:** $C_{\max} = \max(C_1, ..., C_n)$
- **Maximum Lateness:** $L_{\max} = \max(L_1, ..., L_n)$
- **Total weighted completion time:** $\sum w_j C_j$
- **Total weighted tardiness:** $\sum w_j T_j$
- **Weighted number of tardy jobs:** $\sum w_j U_j$

where w_j is the weight (importance) of track j.

At this point, it is possible to formulate the α and β fields of the DSN scheduling problem triplet $\alpha \mid \beta \mid \gamma$ as

$$P_m \mid r_j, \bar{d}_j, brkdwn, M_j \mid \gamma \tag{1.4}$$

Now the difficulty resides in defining the objective function γ. The objective functions presented by Pinedo are always a function of the completion times of the tracks. However, a good DSN schedule does not depend on when the tracks start or end, but how many of the tracks requested are finally scheduled. That means that it is necessary to consider that not all the tracks requests can always be scheduled. That is an oversubscribed scenario.

Pinedo's work about scheduling does not consider the possibility that a job ("track" in this thesis) is not scheduled. For that, binary variables to indicate whether a track has been scheduled or not would be necessary. However, all the algorithms developed in his book cannot be easily modified to introduce these binary variables. Therefore, it is necessary to look for a more general method to formulate and solve the scheduling problem for the DSN.

A more general approach of Pinedo's idea would be directly using **mixed integer linear programming** (MILP). MILP involves optimization problems in which some variables are constrained to be integers while other variables are allowed to be non-integers. The term "linear" refers to both the objective function and the constraints of the problem being linear functions with respect to the problem variables. Using these definitions, the scheduling problem triplet $\alpha \mid \beta \mid \gamma$ can be specialized to the DSN scheduling problem as follows:

$$P_m \mid r_j, \bar{d}_j, brkdwn, M_j \mid \sum w_j z_j \tag{1.5}$$

where z_j is a binary variable indicating whether track j has been scheduled (1) or not (0). Note that now the objective function have to be maximized.

Although the triplets in (1.4) and (1.5) are very similar, the latter optimization problem cannot be solved with the algorithms specific to solve scheduling problems presented in Reference [21]. Instead, the optimization problem represented by triplet (1.5) will be solved using any general method to solve MILP problems (e.g., branch-and-bound, branch-and-cut, heuristic methods, ...). For instance, the MILP solver IBM's CPLEX® uses dynamic search, branch-and-cut and heuristics.

Methods for the assignment problem

The assignment problem for the DSN consists in assigning tracks to antennas. The output of the assignment problem will be one of the inputs for the scheduling problem. For each track, is it necessary to define which antennas or sets of antennas are not eligible for that track. Sets of antennas are necessary for Delta-DOR tracks and tracks using antenna arraying. Also, it is necessary to define a weight w_{ij} between an antenna or set of antennas i and track j which tells how desirable is that assignment. The objective is to assign one antenna or set of antennas to each track in the same

period of time (note that in the assignment problem, an antenna cannot be assigned to different tracks), while maximizing $\sum w_{ij}$ for one option i for each track j.

The assignment problem is a special case of the transportation problem, which is a special case of the minimum cost flow problem, which in turn is a special case of a linear program problem. While it is possible to solve any of these problems using the simplex algorithm, each specialization has more efficient algorithms designed to take advantage of its special structure. Therefore, pros and cons of several methods to formulate and solve the DSN assignment problem are summarized in Table 1.1.

Algorithm	**Pros**	**Cons**
Hungarian method [22]	– Specialized on solving assignment problems – Solves the problem in polynomial time $O(n^3)$ – Handles incompatibilities between antennas and tracks – Easy to implement – The total benefit strictly increases with each iteration	– Difficult to parallelize – Only gives one optimal solution – Does not allow to assign one track to more than one antenna – Maybe too specific to use it in variations of the assignment problem – The problem must be balanced by adding fictitious antennas or tracks
Auction algorithm [23]	– Sometimes faster than Hungarian method for very small problems – Well suited for parallelization. Performs better for problems where data to perform a centralized computation cannot be obtained – Easy to implement – Also solves transportation problems	– The total benefit does not always increase with each iteration – Only gives one optimal solution – Slower for medium and large problems – Requires more memory than Hungarian method

Integer Linear Programming (e.g., branch-and-cut) [24]	– Easy to formulate – Allows variations of the assignment problem – An Integer LP solver can be used – Parallelizable	– Slower than Hungarian method – Not easy to implement – Only gives one optimal solution
Linear Programming (Simplex) [24]	– Easy to formulate – Allows variations of the assignment problem – An LP solver can be used – Parallelizable – Faster than branch-and-cut – While removing the integer constraints, result gives integer values	– Slower than Hungarian method – Not easy to implement – Only gives one optimal solution

Table 1.1: Pros and cons of different methods for the assignment problem

Applying only heuristic algorithms, genetic algorithms or particle swarm optimizations is too slow for the proposed assignment problem, when compared with the algorithms presented in the table above. However, integer linear programming algorithms may be complemented with some heuristics. Also, machine learning is not appropriate for the assignment problem because there is no training data. Finally, checking all possible assignments (i.e., full factorial exploration) in the problem is in most cases impossible due to combinatorial explosion.

The assignment problem could be speedily solved using a specific method such as the Hungarian method or the Auction algorithm. However, the integer linear programming option for the assignment problem could be merged with the mixed integer linear programming option for the scheduling problem, forming a unique MILP problem for the DSN SP, as discussed below.

Merged vs. separate assignment and scheduling problems

At this point, there are the following two possibilities on how to formulate and solve the assignment and scheduling problems:

- **Merged problem:** The assignment and the scheduling problems are combined into a single optimization problem.
- **Separate problems:** The assignment problem is executed before the scheduling problem. The output of the assignment problem is part of the input of the scheduling problem.

The advantages and disadvantages of both options are enumerated in Table 1.2.

Option	Pros	Cons
Merged Problem	– Assignment incompatibilities are avoided as much as possible – Global treatment of "The DSN Scheduling problem"	– Total execution time increases – Complexity of the problem increases – Difficult to implement
Separate Problems	– The assignment problem can be solved with fast algorithms (i.e., Hungarian method) – Both problems are easier to implement when they are separate	– Fast algorithms work only for very specific assignment problems – The output of the assignment problem can generate incompatibilities in the scheduling problem that could have been avoided

Table 1.2: Merged vs. separate assignment and scheduling problems

With this information and after some fast testing, the option of merging both problems in one MILP problem is preferred. This will allow us to implement characteristics of the DSN that would be difficult to incorporate otherwise.

1.6 Thesis Statement

Once MILP has been introduced, it is possible to provide a set of solution-specific research goals that are consistent with the research needs previously identified. These goals are summarized in the following specific problem statement:

To develop an automated long-term scheduling toolkit that satisfies requested DSN tracks both in time and type of resources used **by**:

1. Identifying, characterizing and modeling each requested track
2. Applying the visibility constraints between DSN stations and missions
3. Considering the evolution of the DSN architecture in the following years

4. Developing an algorithm that combines mission needs and DSN resources into a mathematically-formulated problem to solve
5. Solving the mathematical problem that schedules the maximum number of tracks possible given the DSN available resources

using mixed integer linear programming.

1.7 Thesis Structure

The rest of this thesis is structured as follows:

Chapter 2 explains in detail how the automated long-term scheduling toolkit formulates the MILP problem. It first presents some definitions necessary for the following sections. Then it defines the objective function of the optimization problem. Next, the section details the steps involved in formulating the MILP problem. The last section of this chapter mathematically presents the MILP problem.

Chapter 3 presents and analyzes the results of several case inputs, using real NASA missions and DSN inputs. Case 1 replicates a 22.5-year mission several times, imposing hard conditions to the problem. Case 2 simulates a 2-month full DSN scenario. It also discusses computational performance of the toolkit.

Finally, Chapter 4 summarizes the work conducted in this thesis, identifies its main findings and contributions, and highlights opportunities for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

2. Problem Formulation

In order to obtain a schedule of DSN operations for tens of years, there are three important steps: (1) Generating the problem; (2) Solving the problem; and (3) Obtaining the results. In the scope of this thesis, solving the problem is done by an already implemented dynamic search and branch-and-cut provided by IBM's CPLEX® solver. Therefore, generating the problem is the main step to be considered in this chapter, which is divided into the following parts: First, some necessary definitions are presented. Then, the objective function of the problem is defined. Next, the steps involved in formulating the MILP problem are addressed. Finally, the MILP problem is mathematically presented. The software developed in this thesis is named LTST, which stands for Long-Term Scheduling Toolkit.

2.1 Definitions

This section provides some definitions necessary for the rest of the chapter. These definitions complement the concepts presented in Chapter 1.

- **Evolution of the DSN architecture:** Information on the incorporation, upgrade, maintenance and decommission of the antennas of the DSN.
- **Mission:** A mission is a spacecraft, satellite, missile, rover or space station outside Earth. A mission has one or more visibility files associated to it. A mission has several operational segments.
- **Operational segment:** Sequence of tracks of a mission with equal requirements. An operational segment has a start date and an end date. It also defines the desired separation between tracks, the track length, the link requirements and the visibility files to use.
- **Track:** Communication opportunity between a mission and the DSN. A track requested is not always scheduled. Scheduled tracks cannot overlap same resources with other tracks at the same time. A track has technical requirements, a fixed length (usually between 1 and 14 hours) and can move within a fixed window. The window will be reduced by a factor. Each track has a priority associated to it. A track needs one or more links, depending on the type of track.
- **"DSN option":** A "DSN option" is an antenna or combination of antennas (including ground station's array processor if the option uses arraying for downlink), that satisfy the requirements of all links in a given track. Each "DSN option" has an start and an end time depending on the evolution of the DSN architecture.
- **"track option":** A "track option" is a "DSN option" of the track, shortened to a smaller continuous time region that satisfy the visibility requirements and the

time window of the track. A "track option" is active if its track is scheduled in it. It has a MILP binary variable indicating if the option is active. Only zero or one "track option" can be active per track. A "track option" has a MILP continuous variable indicating the start time of the track inside that option. It also has a weight indicating the preference of the option over the rest of the options both from the same track or from different tracks.

- **Visibility:** The visibility of a mission with a ground station is the time intervals when the mission is visible to that ground station. Each time interval starts with a rise time and ends with a set time. A visibility file can be associated to a mission (e.g., a spacecraft) or to a waypoint (e.g., Mars). Visibility constrains the time at which a link can be established.
- **Link:** Connection between a mission and a specific antenna of the DSN (or a specific set of antennas for arraying). Links utilize a given frequency band and typically support a predefined data rate. They can be classified as uplink, a downlink, or both at the same time. Arraying can be used when there is a downlink. In the scope of this thesis, an uplink is mainly constrained by the required Equivalent Isotropically Radiated Power (EIRP), and a downlink by the required gain-to-system noise temperature ratio (G/T) of the antenna (DSS).
- **Equivalent isotropically radiated power (EIRP):** "Product of transmitter power and antenna gain in a given direction relative to an isotropic antenna of a radio transmitter" [25]. This is the minimum power that a DSN antenna must radiate in order for the mission to receive the uplink signal.
- **Gain-to-system noise temperature ratio (G/T):** "Figure of merit in the characterization of antenna performance, where G is the antenna gain in decibels at the receive frequency, and T is the equivalent noise temperature of the receiving system in kelvins. The receiving system noise temperature is the summation of the antenna noise temperature and the RF chain noise temperature from the antenna terminals to the receiver output" [26]. To receive a downlink signal, a DSN antenna must provide a minimum G/T so that the link budget closes.

Some concepts presented above are broadened throughout the rest of the chapter.

2.2 Objective Function

To solve the DSN SP, the following objective function is considered:

$$\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \sum_{\boldsymbol{j=1}}^{\boldsymbol{M_i}} (\boldsymbol{w_{ij}} \times \boldsymbol{a_{ij}}) \tag{2.1}$$

where w_{ij} is the weight of "track option" j of track i, a_{ij} is a binary variable that indicates whether that option is active or not, N is the number of tracks in the problem, and M_i is the number of "track options" for track i. Note that constraints

explained later will guarantee that no more than one "track option" per track is active.

Since all weights are greater than zero, the MILP algorithm tries to set to 1 as many a_{ij} variables as possible, thus trying to maximize the number of tracks scheduled. However, each "track option" has its own weight which indicates the preference of choosing that option over all others. The way in which these weights are computed is addressed below.

2.2.1 Weight of a "track option"

Weights in objective function (2.1) act in several ways. On the one hand, weights guide the MILP algorithm during the optimization process. Particularly, the MILP algorithm will generally try to start setting to 1 the a_{ij} variables with higher weights. On the other hand, weights, together with constraints, specify the optimal solution of the MILP problem.

It is worth explaining how weights are assigned to "track options", how they influence in the evolution of the MILP algorithm and how they impact the solution of the problem by distinguishing two cases: Weights of "track options" of different tracks and weights of "track options" of the same track. These are separately discussed in the following two sections.

Weights of "track options" of different tracks

The difference between weights of "track options" of different tracks determine which tracks are finally scheduled in oversubscribed intervals. Those weights also influence in the order in which the MILP algorithm handles each track.

Four different values for w_{ij} –weight of "track option" j of track i– are initially tested:

$$w_{ij} = 1 \tag{2.2a}$$
$$w_{ij} = l_i \tag{2.2b}$$
$$w_{ij} = \log(1 + l_i) \tag{2.2c}$$
$$w_{ij} = 1 + \log(1 + l_i) \tag{2.2d}$$

where l_i is the length of track i in hours. The tests were run using a test input data set that resulted in a dense but not oversubscribed schedule. Therefore, the solutions obtained with the different w_{ij} alternatives were equally good. Then, the study focused on how fast was each form of w_{ij}. Assigning the same weight (e.g., 1) to all tracks was somewhat slow, because the MILP algorithm did not differentiate between any tracks. Therefore, we moved to more heuristic-based weight, where its value was positively correlated with track length. Indeed, it is probably easier to first schedule longer tracks and then fill the gaps with shorter tracks (instead of the opposite or random direction).

Using this idea, we first tried to assign a weight equal to the track length. However, that eliminated most shorter tracks because the difference in lengths of different tracks could be an order of magnitude or higher. Therefore, applying the logarithm

as shown in Equation (2.2c) put the weights on a similar level but with different values. This resulted in a total solver execution time that was faster than the other two previous options. A 1 added to the track length is to avoid negative weights when the logarithm is applied. Finally, Equation (2.2d) ended up being the fastest by far. Adding a constant value to the logarithm made the MILP algorithm tend to consider that not only the lengths of the scheduled tracks were important, but the total number of scheduled tracks.

On the other hand, oversubscribed scenarios are considered as more common in DSN operations. Consequently, the weight of any given track should mainly depend on its relative importance with respect to all other tracks, rather than just focusing on MILP speed. Therefore, in the latest version of the scheduling toolkit the weight of a "track option" is initially defined as

$$w_{ij} = \left(1 + \frac{s_i}{l_i}\right)\left(1 + \frac{\log\left(1 + l_i\right)}{2}\right) \tag{2.3}$$

where l_i is the length of track i and s_i is the desired separation between tracks i and $i+1$, both values in hours. The idea behind Equation (2.3) is that tracks that cover less proportional time within their operational segment have preference over the others. For example, if there is an operational segment with 6-hour tracks and a separation of about 6 hours between tracks, and another operational segment also has 6-hour tracks and a separation of about 1 month, tracks of the later operational segment will have higher preference (higher weight) than tracks of the former operational segment. This is due to the fact that not scheduling a track that occurs once per month is much worse than not scheduling a track that normally occurs twice a day. On the other hand, the logarithmic dependency with the length of the track still slightly increases the preference for longer tracks.

Additionally, some tracks require a hot backup. This feature will be detailed later in this thesis. However, a track with a hot backup requirement is a track with maximum priority, such as human missions. Therefore, weights of "track options" of this type of tracks are heavily increased. These weights are finally 10,000 times the value obtained from Equation (2.3), which ensures that tracks with hot backup are always scheduled.

Weights of "track options" of the same track

Different weights are assigned to "track options" of the same track. Since up to one "track option" per track can be active, the MILP algorithm gives preference to choose an option with higher weight for each track. The key is that the "track option" chosen for a track disturbs as less as possible other tracks. To achieve this, higher weights are assigned to "track options" that:

- **Use less amount of resources:** A "track option" that uses less resources compared to the rest of options of the same track disturbs less to other tracks. This is because it keeps available more resources for other tracks. For example, an option using only one 70-meter antenna has a higher preference (higher weight) than an option for the same track arraying three 34-meter antennas (and the array processor) in the same ground station. Therefore, using only one

the 70-meter antenna keeps available antennas for up to three other missions in that ground station. Not only the number of resources is important, but the maximum capacity and importance of each resource. For instance, an option using one 34-meter antenna has a higher preference (higher weight) than an option for the same track using one 70-meter antenna because then, the 70-meter antenna still is available for another track that cannot happen in a 34-meter antenna. Another example is when an option does arraying, it uses the array processor of the ground station. Since there is currently one array processor per ground station, that option does not allow other tracks to do arraying in the same ground station simultaneously, even when the antennas required for the arraying are available. Therefore, an option using the array processor of a ground station is generally assigned a lower weight.

- **Have a larger window:** Options with a large window compared to the length of the track allow the track to be easily rescheduled without changing "track option". This fact does the option less disturbing to other tracks, as its track is more flexible in time. This option has a higher weight than the others of the same track. Opposite, a "track option" where the window is slightly larger than the length of the track makes that track difficult to move, so it disturbs other tracks more.

A combination of the two previous concepts defines the final weight assigned to a "track option" in comparison to the rest of "track options" of the same track. Weights of "track options", initially defined as in Equation (2.3), are slightly modified applying these concepts. Note that the differences between weights of "track options" of the same track are kept small. Particularly, those differences are orders of magnitude less than the differences between "track options" of different tracks. This is because the concepts presented in this section are secondary objectives of the problem. Priority is given to scheduling as much tracks (the ones with higher preference) as possible.

2.3 Problem Formulation Process Flow

The software, programmed in Python, starts reading several input files: The DSN architecture evolution file, the missions scheduler file, and the visibility files. The scheduler files contains the operational segments for all the missions. The user can specify a start and end dates, so only tracks within the specified region will be simulated. Operational segments of the missions in the scheduler input file are added to the simulation. The data from the input file for an operational segment read is its id and name, the start and end dates and times of the operational segment, the link type, the frequency band, the days between tracks, the track length, whether hot backup is required or not, the associated visibility information, the link direction(a), and the G/T and/or EIRP values. With this information it is possible to generate the tracks of the operational segment.

2.3.1 Global generation algorithm

The following pseudo-code summarizes the process flow of the DSN scheduling toolkit:

Algorithm 1: Generation algorithm of the LTST

```
input: scheduler S, visibility V, DSN evolution E
output: MILP problem
begin
  problem ← ∅
  foreach op_segment in S
    Compute 'DSN options' using op_segment and E

    foreach track in op_segment
      Compute track limits
      Compute 'track options' using 'DSN options' and V
         and track limits
      track ← track + 'track options'
      foreach 'track option' in track
        problem ← var active
        problem ← var t_start
        Compute compatibilities of 'track option'
        foreach compatibility in compatibilities
          if compatibility is fully compatible
            continue

          problem ← var both_active
          problem ← constraint both_active

          if compatibility is partially compatible
            if compatibility is double-constrained
              problem ← var before
              problem ← var after
              problem ← constraint before and after
              problem ← constraint avoid overlap #1
              problem ← constraint avoid overlap #2
            else if compatibility is single-constrained
              problem ← constraint avoid overlap
            end
          else if compatibility is incompatible
            problem ← constraint limit 'track options'
          end
        end
      end
      problem ← constraint limit 'track options'
    end

  end

  return problem
end
```

Each of the steps are explained in detail throughout this chapter. The first step is computing the "DSN options" for the operational segment as explained below.

2.3.2 "DSN options"

The "DSN options" associated to an operational segment are all the possible combinations of resources (antennas) that satisfy the given link requirements. As all tracks in an operational segment have the same properties (e.g., frequency band, EIRP...) the "DSN options" only needs to be computed once per operational segment. "DSN options" also take into account the evolution of the DSN, by indicating start and end dates of the option.

Three types of link can be established: uplink, downlink and uplink&downlink. Also, Delta-DOR tracks, which do two simultaneous downlinks to two antennas in different ground stations, must be treated separately. Explanations about computing the "DSN options" for uplink, downlink, uplink&downlink and Delta-DOR are separately addressed below. After, the special case of links requiring hot backup is explained.

Each individual link is defined by a *link equation* or *link budget* which relates all of the parameters needed to compute the signal-to-noise ratio of the communications system. The basic equation used in sizing a wireless communication link is

$$\frac{E_b}{N_0} = \frac{PL_lG_tL_sL_aG_r}{kT_sR} \tag{2.4}$$

where E_b/N_0 is the ratio of received energy-per-bit to noise-density, P is the transmitter power, L_l is the transmitter-to-antenna line loss, G_t is the transmit antenna gain, L_s is the space loss, L_a is the transmission path loss, G_r is the receive antenna gain, k is the Boltzmann's constant, T_s is the system noise temperature, and R is the data rate. The propagation path length between transmitter and receiver determines L_s, while L_a is a function of factors such as rainfall density. For deep space missions, E_b/N_0 is in range -0.10 dB to 5 dB [27].

The term PL_lG_t can be replaced by the $EIRP$, in watts. Also, Equation (2.4) can be rewritten in decibels as

$$E_b/N_0 = EIRP + L_s + L_a + G_r + 228.6 - 10\log T_s - 10\log R \tag{2.5}$$

where the $EIRP$ is in dBW, E_b/N_0, L_s, L_a and G_r are in dB, T_s is in K, R is in bps and $10\log k = -228.60$ dBW/(Hz·K). The sensitivity of the receiving station is defined as $G_r/T_s = G_r - 10\log T_s$ and is expressed in dB/K. In each link, L_s, L_a, R, and the minimum value of E_b/N_0 are considered fixed and known information. Therefore, the $EIRP$ of the transmitter and the G_r/T_s of the receiver determines if the E_b/N_0 obtained using Equation (2.5) satisfies a minimum acceptable value.

For the purposes of this thesis, the input file indicates the minimum $EIRP$ of the transmitter DSN antenna required for uplinks and the minimum G_r/T_s of the receiver DSN antenna(s) required for downlinks.

Uplink options

Computing all the "DSN options" for an uplink is simple since uplinks only require one antenna. For each antenna, the intervals –within the limits of the operational segment– when the antenna is capable of supporting the requested uplink are computed. A link is supportable by a given antenna if it is in view with spacecraft for a long enough period of time, it supports the desired frequency band and meets the EIRP requirement specified in the input file.

Example 1. Some of the "DSN options" for an operational segment (year 2005) requiring Ka32-band uplink tracks are:

- *"DSN option 1.1"* uses DSS-34 in Canberra
- *"DSN option 1.2"* uses DSS-24 in Goldstone
- *"DSN option 1.3"* uses DSS-65 in Madrid (from 2005-1-1 to 2005-7-14)*
- *"DSN option 1.4"* uses DSS-65 in Madrid (from 2005-7-30 to 2005-12-31)*

* DSS-65 in maintenance from 2005-7-15 to 2005-7-29.

Downlink options

In order to compute the "DSN options" for a downlink track that uses only one antenna, a very similar procedure to the one explained for the uplink options is followed. The difference is that in this case the antenna must support a downlink connection in the specified frequency band during the interval. Now, to establish and maintain a downlink connection it is necessary that the antenna provides a G/T greater or equal to the requirement set by the operational segment.

On the other hand, in order to obtain the downlink "DSN options" for a track that requires antenna arraying, almost all combinations of antennas that support downlink in the specified frequency band are generated. Each combination must have between two and four antennas from the same ground station. Also, the array processor of the ground station is added to the resources used by the combination. Finally, the G/T required by the track must be lower or equal to the one provided by the array, which is calculated as

$$\mathrm{G/T_{array}(dB)} = 10 \times \log\left(\eta_{\mathrm{array}} \cdot \sum_{i \in \mathrm{array}} 10^{\frac{\mathrm{G/T}_i\mathrm{(dB)}}{10}}\right) \tag{2.6}$$

where $\mathrm{G/T}_i$ (dB) is the G/T in decibels of antenna i in the array, and η_{array} is an efficiency factor of the arraying system. In the current version of the LTST, η_{array} is always 70%. If the G/T of the array satisfies the minimum G/T required by the mission, then that array becomes a valid "DSN option" for the downlink connection. However, the combinations where a subset of its resources is already a valid "DSN option" for the downlink connection, are discarded. For example, if an array of two antennas already satisfy the minimum G/T requested in the specified frequency band, any array of more antennas containing these two antennas is discarded.

Example 2. Some of the "DSN options" for an operational segment (year 2007) requiring X-band downlink tracks are:

- *"DSN option 2.1"* uses DSS-34, DSS-35 and the array processor of Canberra
- *"DSN option 2.2"* uses DSS-54, DSS-55 and the array processor of Madrid
- *"DSN option 2.3"* uses DSS-54, DSS-65 and the array processor of Madrid
- *"DSN option 2.4"* uses DSS-43 (70-meter antenna) in Canberra
- *"DSN option 2.5"* uses DSS-24, DSS-25 and the array processor of Goldstone (from 2007-1-1 to 2007-2-19)*
- *"DSN option 2.6"* uses DSS-24, DSS-25 and the array processor of Goldstone (from 2007-3-11 to 2007-12-31)*

* DSS-24 in maintenance from 2007-2-20 to 2007-3-10.

Uplink&downlink options

To obtain uplink&downlink options, the downlink options using the same link requirements are computed first. Now, for each downlink option it is necessary to compute the intervals when one or more of its antennas support the requested uplink (minimum EIRP in the specified frequency band) to the mission. Those uplink "DSN options" must be within the time frame of the downlink option and use an antenna used by the downlink option. Furthermore, the union of the intervals of the specific uplink "DSN options" associated to the same downlink option is computed. The resulting time intervals, using the resources of its associated downlink option, are the "DSN options" for the uplink&downlink connection.

Example 3. Some of the "DSN options" for an operational segment (year 2007) requiring X-band uplink&downlink tracks are exactly the same options as the example presented for downlink tracks (Example 2). This applies because each of those options has at least one antenna that support the X-band uplink requested.

Delta-DOR options

A Delta-DOR track requires establishing and maintaining two downlinks to different ground stations simultaneously. Each of these downlinks can only use one 34-meter antenna. Therefore, all combinations of two 34-meter antennas of different ground stations are first generated. Then, for each combination, time intervals when both antennas can continuously support a downlink (minimum G/T in the specified frequency band) to the mission are the "DSN options" for the Delta-DOR track.

Example 4. Some of the "DSN options" for an operational segment (year 2008) requiring X-band Delta-DOR tracks are:

- *"DSN option 4.1"* uses DSS-54 (Madrid) and DSS-35 (Canberra)
- *"DSN option 4.2"* uses DSS-34 (Canberra) and DSS-55 (Madrid)
- *"DSN option 4.3"* uses DSS-24 (Goldstone) and DSS-54 (Madrid)
- *"DSN option 4.4"* uses DSS-25 (Goldstone) and DSS-54 (Madrid)
- *"DSN option 4.5"* uses DSS-35 (Canberra) and DSS-25 (Goldstone)

Options with hot backup requirement

The hot backup requirement is set to all tracks of human missions, which have maximum priority with respect to the scheduling problem. This requirement implies that, throughout the duration of the track, it is necessary to have extra resources available (e.g., antennas) to reestablish and maintain the track in case one resource fails.

In order to compute "DSN options" with the hot backup requirement, the options without considering the requirement should be obtained first. With the original "DSN options", all combinations of two options are generated. Then, the combinations where any resource (antenna or array processor) is used by the two original options are discarded. For each of the remaining combinations, the intersection (if any) of the intervals of the two original options is a "DSN option" with hot backup requirement. Therefore, the resources assigned to this option are the combination of resources used by the two original options.

Example 5. Consider only the six "DSN options" listed in the example for downlink tracks (Example 2). When the hot backup requirement is added, thirteen possible "DSN options" with hot backup requirement are obtained. These are all the possible combination (fifteen) of two of the six original options, except for two combinations: options 2.5 and 2.6 –because they do not overlap in time–, and options 2.2 and 2.3 –because some resources are used by both options–. Some of the "DSN options" with hot backup requirement are:

- *"DSN option 2.1+2.2"* uses DSS-34, DSS-35, DSS-54, DSS-55 and the array processors of Madrid and Canberra
- *"DSN option 2.1+2.4"* uses DSS-43, DSS-34, DSS-35 and the array processor of Canberra
- *"DSN option 2.3+2.5"* uses DSS-54, DSS-65, DSS-24, DSS-25 and the array processors of Goldstone and Madrid (from 2007-1-1 to 2007-2-19)

2.3.3 "Track options"

Once the "DSN options" for a given operational segment have been computed, the software creates each track one by one. The first step for each track is to determine its time window. The number of tracks in an operational segment is approximately obtained by dividing the length of the operational segment by the sum of the track length and the separation between consecutive tracks. Then, the time window of the operational segment is divided by the number of tracks it has, and this division is finally reduced by a certain margin to obtain the time window of a given track. Figure 2-1 illustrates this process.

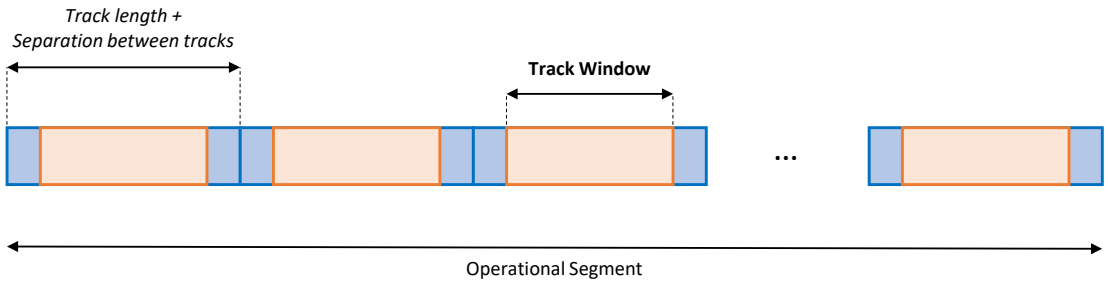

Figure 2-1: Division of track windows in an operational segment

The reduction margin applied to a track is the same for all tracks in the same operational segment. The margin is currently calculated as the factor

$$m_i = 0.95 \left(\frac{s_i}{s_i + l_i} \right)^8 \tag{2.7}$$

The length of a division in the operational segment $(s_i + l_i)$ is reduced by m_i (is multiplied by $1 - m_i$) and then centered, to obtain the time window of the track. In case that the time window was shorter than the track length, it is enlarged to the track length. The goal of Equation (2.7) is to apply a bigger reduction margin when there is a lot of separation between tracks, and less reduction margin when the separation is less. By doing this, tracks that would initially have a time window of months, end up having a time window of a week, which greatly reduces the computational complexity of the MILP solver.

At this point, all "track options" for a given track are created. Each operational segment has visibility information associated. Each "DSN option" of the operational segment uses resources from one or more ground stations. The "track options" of a track are calculated by intersecting (in time) each "DSN option" of the track's operational segment with the correspondent visibility information and with the time window of the track. Note that each "DSN option" must be intersected with the visibility information of each ground station it uses. "track options" with a length less than the track length are discarded. This process is illustrated in Figure 2-2.

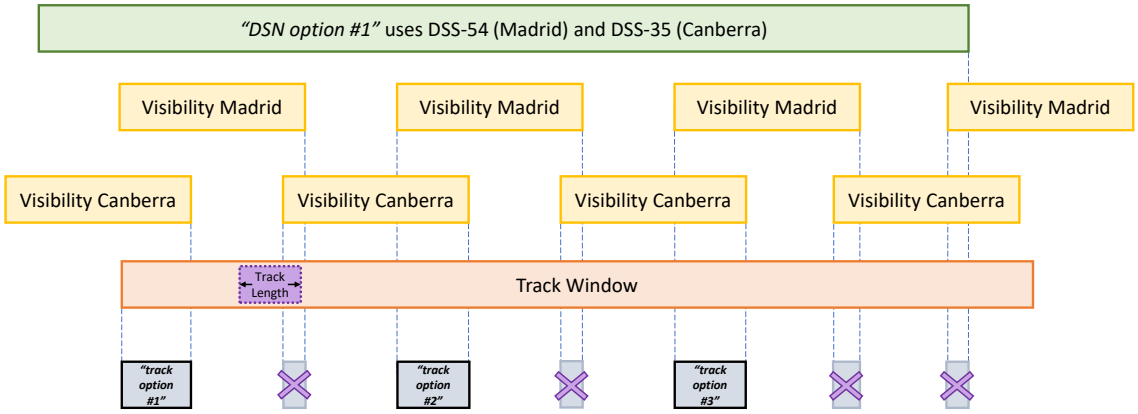

Figure 2-2: Creation of "track options" for a Delta-DOR track, using one "DSN option", visibility information and track window

To sum up, "track options" are computed from *"DSN options"* + *visibility input* + *track limits*. See the following example.

Example 6. Consider only the first two "DSN options" listed in the example for uplink tracks (Example 1). When the track window is February 22-23, 2005, "track options" are:

- *"track option 6.1"* uses *"DSN option 1.1"* from 2005-2-22 10:04 to 14:27 (visibility in Canberra)
- *"track option 6.2"* uses *"DSN option 1.1"* from 2005-2-23 16:59 to 20:27 (visibility in Canberra)
- *"track option 6.3"* uses *"DSN option 1.2"* from 2005-2-22 11:04 to 15:27 (visibility in Goldstone)
- *"track option 6.4"* uses *"DSN option 1.2"* from 2005-2-22 19:01 to 22:34 (visibility in Goldstone)

Created variables

For each "track option", the following two variables are added to the MILP problem:

- **One binary variable:** It indicates whether the "track option" is active (1) or not (0). Its initial value is set to 0 (not active).
- **One continuous variable:** It indicates the start time of the track within the "track option" if scheduled in that option. Its lower bound is the start time of the "track option" and its upper bound is the end time minus the length of the track. Its initial value is set to a random number between its upper and lower bounds.

Finally, one constraint for the binary variables associated to the same track is added to the MILP problem. This constraint ensures that no more than one "track option" is active per track.

2.3.4 Compatibility between tracks

Scheduled tracks must not overlap. In order to guarantee this, we add one or multiple constraints to the MILP problem depending on their *compatibility*. Three types of track compatibility are defined: fully compatible, partially compatible, and incompatible. The algorithm currently implemented is optimized to compare only some specific "track options", as most are fully compatible. Each type of compatibility is individually explained below.

Fully compatible "track options"

Two "track options" are **fully compatible** if an option does not interfere with the other. This happens if any of these three conditions apply:

- Both "track options" do not overlap in time.
- There is no resource used by both "track options".
- Both "track options" correspond to the same track, because there is already the constraint that only one "track option" can be active per scheduled track.

Figure 2-3 shows two fully compatible tracks. They are fully compatible because whenever both tracks are scheduled within these "track options", they never overlap.

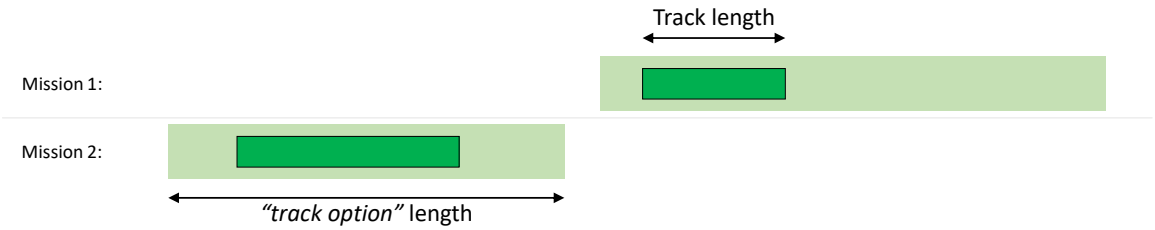

Figure 2-3: Two fully compatible "track options"

Partially compatible "track options"

Two "track options" are **partially compatible** if an option partially interferes with the other. This happens for all non-fully compatible pairs that can schedule both tracks without overlapping. Figure 2-4 shows two partially compatible tracks. They are partially compatible because, depending on when both tracks are scheduled within these "track options", they overlap or not.

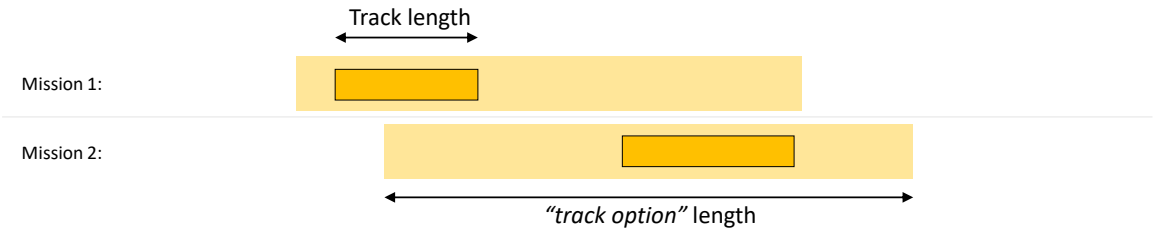

Figure 2-4: Two partially compatible "track options"

When the software finds a partially compatible pair of "track options", the necessary constraints are added to the MILP problem to ensure that both tracks do not overlap (if both "track options" are active).

Incompatible "track options"

Two "track options" are **incompatible** if an option fully interferes with the other. This happens for all non-fully compatible pairs that cannot schedule both tracks without overlapping. Figure 2-5 shows two incompatible tracks. They are incompatible because whenever both tracks are scheduled within these "track options", they always overlap.

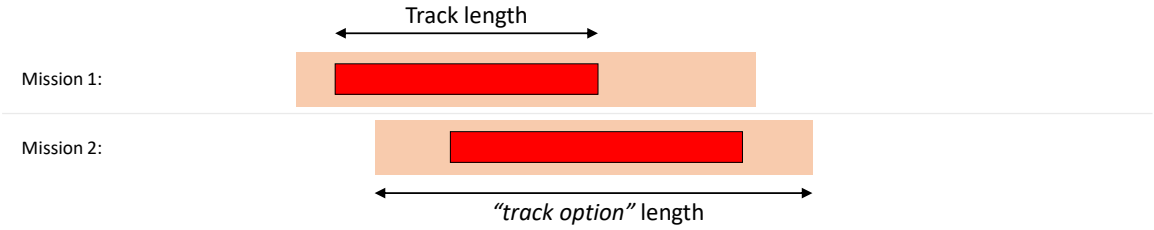

Figure 2-5: Two incompatible "track options"

When the software finds an incompatible pair of "track options", one constraint is added to the MILP problem to ensure that no more than one of the two "track options" is active.

2.4 The MILP Problem

Throughout this chapter, the process flow to formulate the MILP problem has been explained in detail. This section mathematically presents the whole MILP problem. Afterwards, constraints created due to compatibility issues between "track options" are separately explained.

2.4.1 Optimization problem

The mathematical formulation of the optimization problem –MILP problem– of the scheduling toolkit of this research is

$$\underset{a,\, t_{start}}{\text{maximize}} \quad \sum_{i=1}^{N}\sum_{j=1}^{M_i}(w_{ij} \times a_{ij}) \tag{2.8a}$$

$$\text{subject to} \quad \sum_{j=1}^{M_i} a_{ij} \leq 1, \qquad \begin{matrix} i = \{1,\ldots,N\}, \\ j = \{1,\ldots,M_i\} \end{matrix} \tag{2.8b}$$

$$\text{Constraints for partial compatibility} \tag{2.8c}$$

$$\text{Constraints for incompatibility} \tag{2.8d}$$

$$a_{ij} \in \{0,1\}, \qquad \begin{matrix} i = \{1,\ldots,N\}, \\ j = \{1,\ldots,M_i\} \end{matrix} \tag{2.8e}$$

$$t_{min_{ij}} \leq t_{start_{ij}} < t_{max_{ij}} - l_i, \qquad \begin{matrix} i = \{1,\ldots,N\}, \\ j = \{1,\ldots,M_i\} \end{matrix} \tag{2.8f}$$

where a_{ij} and $t_{start_{ij}}$ are the variables of the problem. Formula (2.8a) is the same objective function as in (2.1). N is the number of tracks in the problem, and M_i is the number of "track options" of track i. w_{ij} (real numbers greater than zero) are the weight of "track option" j of track i computed as explained in Section 2.2.1. Binary variables a_{ij} represent whether "track option" j of track i is active (value 1) or not (value 0). Continuous variables $t_{start_{ij}}$ indicate the start time of track i if it is scheduled in "track option" j of track i. $t_{start_{ij}}$ is lower bounded by the start time ($t_{min_{ij}}$) of "track option" j of track i. Similarly, it is upper bounded by the end time ($t_{max_{ij}}$) of "track option" j of track i minus the length of track i (l_i). Constraint (2.8b) ensures that no more than one "track option" per track is active. Finally, Constraints (2.8e) and (2.8f) are created when two "track options" are non-fully compatible. These constraints are explained below.

2.4.2 Constraints for partially compatible "track options"

Consider that "track option" j_1 of track i_1 and "track option" j_2 of track i_2 are partially compatible. The following set of variables and constraints need to be added in order to guarantee that both tracks do not overlap if both "track options" are active. First, a new binary auxiliary variable $a_{(ij)_1(ij)_2}$ is added to the MILP problem. This binary variable is defined using an *indicator constraint*, an efficient implementation of the Big M method in IBM's CPLEX® for "activating" or "deactivating" constraints. This variable is defined as the following *if-only-if indicator constraint*

$$a_{(ij)_1(ij)_2} = \begin{cases} 1, & \text{if } a_{(ij)_1} + a_{(ij)_2} = 2 \\ 0, & \text{otherwise} \end{cases} \tag{2.9}$$

It indicates whether both "track options" are active (value 1) or not (value 0).

Constraints to ensure that track i_1 and track i_2 do not overlap when $a_{(ij)_1(ij)_2}$ is set to 1 must also be added. Two separate cases are considered below.

Double constraints

In this case, two tracks could be scheduled one before the other. Two new binary auxiliary variables, $be_{(ij)_1(ij)_2}$ and $af_{(ij)_1(ij)_2}$, are added to the MILP problem. When $be_{(ij)_1(ij)_2}$ is set to 1, it indicates that track i_1 is done before track i_2. On the other hand, when $af_{(ij)_1(ij)_2}$ is set to 1, then track i_1 is done after track i_2. These binary variables are restricted using the following *if indicator constraint*

$$a_{(ij)_1(ij)_2} = 1 \implies be_{(ij)_1(ij)_2} + af_{(ij)_1(ij)_2} = 1 \tag{2.10}$$

This ensures that when $a_{(ij)_1(ij)_2}$ is set to 1, one and only one of variables $be_{(ij)_1(ij)_2}$ and $af_{(ij)_1(ij)_2}$ is set to 1.

Finally, the following 'double' *if indicator constraints* are added to the problem

$$be_{(ij)_1(ij)_2} = 1 \implies t_{start_{(ij)_2}} - t_{start_{(ij)_1}} \geq l_{i_1} \tag{2.11a}$$
$$af_{(ij)_1(ij)_2} = 1 \implies t_{start_{(ij)_1}} - t_{start_{(ij)_2}} \geq l_{i_2} \tag{2.11b}$$

Constraints (2.11a) and (2.11b) ensure that track i_1 and track i_2 do not overlap.

Single constraints

'Single' constraints are added instead of 'double' for two partially compatible tracks when it is known that only one of the tracks could execute before the other. This can happen depending on the time window of the two "track options" and the length of both tracks, a situation that is detected when generating the MILP problem. In this case, no $before$ or $after$ variables are created. Instead, one of the following 'single' *if indicator constraints* is added to the problem

$$a_{(ij)_1(ij)_2} = 1 \implies t_{start_{(ij)_2}} - t_{start_{(ij)_1}} \geq l_{i_1} \tag{2.12a}$$

OR

$$a_{(ij)_1(ij)_2} = 1 \implies t_{start_{(ij)_1}} - t_{start_{(ij)_2}} \geq l_{i_2} \tag{2.12b}$$

Constraint (2.12a) is added to the problem if track i_1 can only be done before track i_2. Similarly, Constraint (2.12b) is added to the problem if track i_1 can only be done after track i_2.

2.4.3 Constraints for incompatible "track options"

Consider that "track option" j_1 of track i_1 and "track option" j_2 of track i_2 are incompatible. Then, the following constraint is added to the MILP problem

$$a_{(ij)_1} + a_{(ij)_2} \leq 1 \tag{2.13}$$

Since tracks in these "track options" would always overlap, Constraint (2.13) implies that not both "track options" are active. This ensures no overlapping of the tracks.

2.4.4 Performance optimizations

The toolkit is implemented in Python and uses IBM's CPLEX® to solve the MILP problem. The solver uses dynamic search, branch-and-cut, and sometimes applies heuristic algorithms during the branch-and-cut. It also provides *indicator constraints*, which are used instead of Big M method constraints. Finally, it also allow to define a time limit and an optimality level, that when reached the execution stops, and the best result achieved is provided.

The performance of the solver depends on its configuration, and heavily on the problem to solve. Therefore, some changes are done to the original algorithm presented in Section 2.3.1, in order to improve computational performance. The two relevant changes are explained below:

- After some testing of the toolkit, it was found that there were tracks with more than 100 "track options", which was unnecessary and slowed down the performance. In fact, with just a few "track options" per tracks is enough to get a good solution. Therefore, now "track options" are limited to a maximum of 8 per track, which are randomly chosen from the 18 "track options" with highest weight in the track. Other "track options" are discarded. These values were chosen after some testing. However, they are customizable.
- When the MILP problem becomes very big, the toolkit executes the solver periodically as new batches of variables and constraints are added to the problem. This strategy is analogous to a greedy algorithm: After the solver is executed, constraints fixing all the integer (binary) variables to the solution are added to the problem. Therefore, the solver only optimizes the new tracks added between the previous and the current execution of the solver. However, when the problem is fully formulated, all constraints fixing variables are removed. Then, the solver is executed starting with the advanced feasible solution previously obtained.

Additionally, some minor tuning of the solver's configuration parameters is done.

THIS PAGE INTENTIONALLY LEFT BLANK

3. Results

In this chapter, two cases run with the developed DSN schedule toolkit are presented. The first case deals with the whole input of one mission. This mission lasts 22 and a half years approximately. This mission is replicated up to 10 times to have a set of several missions for the full 22.5 years. The second case corresponds to a 2-month full real DSN scenario with 34 different missions. Each case is explained in a section below.[3]

3.1 Case 1: Single mission replicated N times

The base input data for this case corresponds to a real NASA mission lasting 22.5 years. There are also two visibility input files: one pointing to the mission and the other to a waypoint. The DSN architecture evolution file provided as input is the current DSN architecture without any change during all period of interest. The mission has 100 operational segments that generate 12,556 tracks. The mission is then replicated up to 10 times in order to see how the software schedules missions asking for the same resources at the same moment. First, results are analyzed, and then, memory and time performance of the simulations are commented.

3.1.1 Analysis of the results

Table 3.1 shows some general results of this problem. Multiples simulations using 1 to 10 missions (same mission replicated) are executed. The first item to highlight from the table is that not all tracks are scheduled when there is only one mission. About 10% of the tracks are not scheduled in this test. With one mission it is possible to expect that all tracks are scheduled, but there are two reasons of why this is not true in our test. First, there are tracks that do not have any "track option". This is typically due to visibility restrictions rather than link requirements. The input can request some tracks that are slightly larger than the visibility window for that period of the year. Therefore, it is completely impossible to schedule that track even if there were no more tracks in the system. And second, the input of a mission can have overlapping operational segments, specially when they are of different type.

Another point to highlight from the information on the table is that the percentage of tracks scheduled considerably decreases as the number of missions increases. Since all missions are identical, their "track options" are also identical. Therefore, they are requesting the same set of resources in the same period of time. The time window of the options allows some flexibility to schedule tracks in the same period of time.

[3] Author's Note: Simulations on this chapter have been executed using past, present and future NASA missions and DSN information. In order to obtain the clearance for unlimited release of this thesis, several information and details of these simulations have been intentionally excluded from this chapter.

However, the amount of resources of the DSN is limited. Therefore, the more tracks requested, the more tracks not scheduled.

#missions	#operational segments	#tracks	#hours tracks	#scheduled tracks	#hours scheduled tracks
1	100	12,556	99,432	11,598	91,602
2	200	25,112	198,865	20,027	157,753
3	300	37,668	298,297	28,359	223,085
4	400	50,224	397,730	36,664	288,212
5	500	62,780	497,162	44,499	349,533
6	600	75,336	596,594	52,345	410,918
7	700	87,892	696,027	60,178	472,171
8	800	100,448	795,459	68,005	533,404
9	900	113,004	894,892	75,652	593,192
10	1000	125,560	994,324	83,306	653,052

Table 3.1: General results (Case 1)

Figure 3-1 plots the percentage of tracks and percentage of hours scheduled when run with 1 to 10 missions. As expected, it shows a decreasing trend in the percentages. For instance, simulations with 6 or more missions schedule less than 70% of the tracks requested. Percentages of hours scheduled are almost equal to its corresponding percentage of scheduled tracks, because tracks generally have similar lengths.

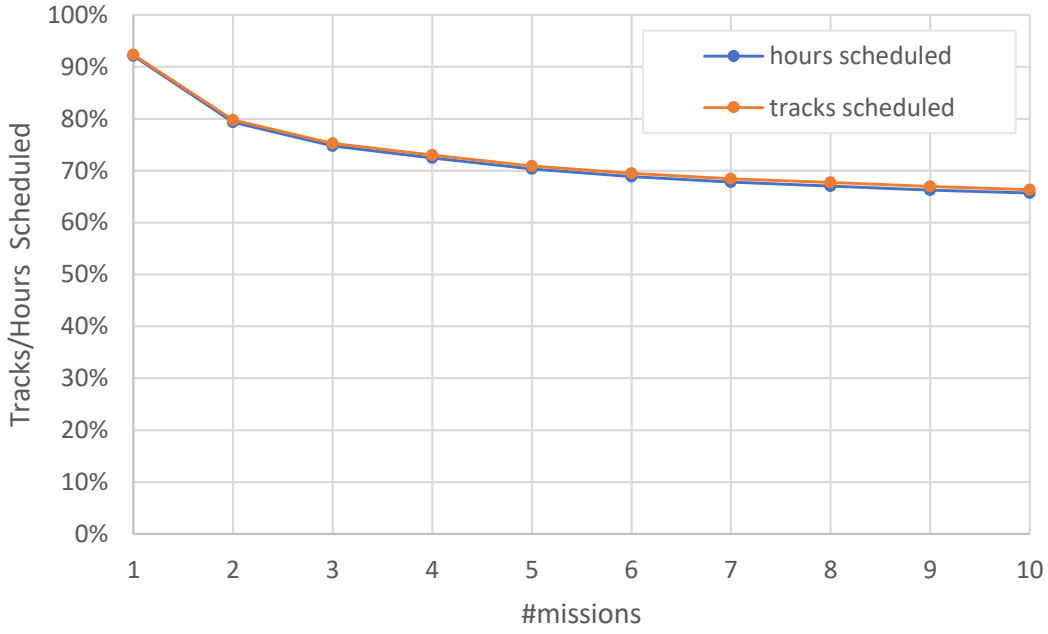

Figure 3-1: Tracks/hours scheduled vs. number of missions (Case 1)

3.1.2 Computational performance

Computational performance is kept to acceptable levels (i.e., a few minutes) thanks to the performance optimizations presented in Section 2.4.4. Figure 3-2 plots the execution time of each run, which shows that the execution time tens to increase non-linearly with the number of input missions.

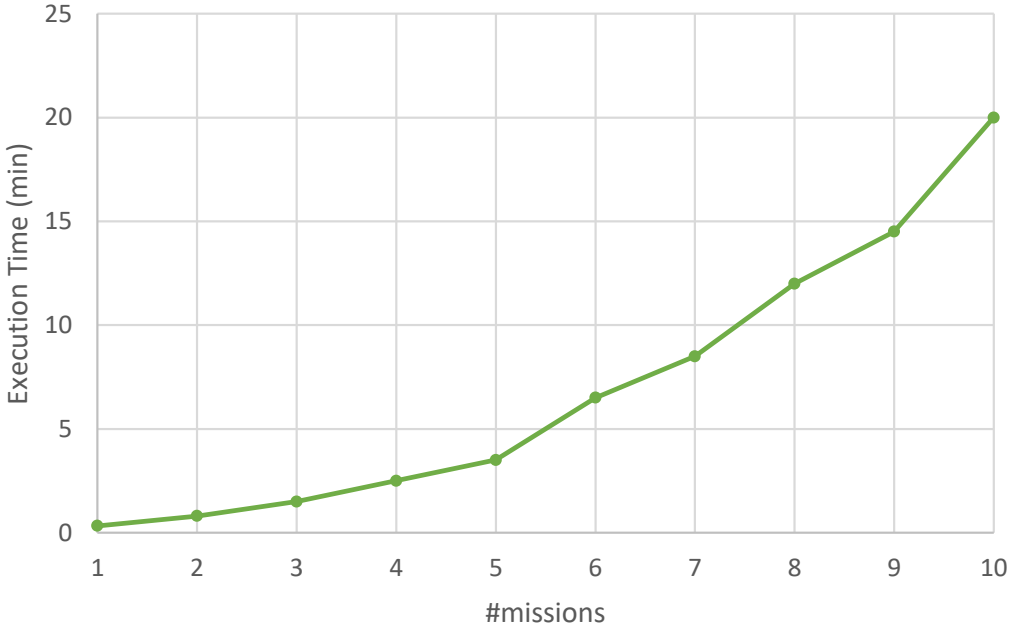

Figure 3-2: Time performance (Case 1)

The computer memory used by the simulations has a similar behavior as time performance. The more missions, the more memory used. For example, the complete MILP problem when simulating 10 missions replicated has more than 1.3 million variables and more than 4.1 million constraints. Generally, the more variables and constraints has the problem, the more memory will be used by the toolkit and the solver. Similarly, the more time IBM's CPLEX® is running, the more memory is used. That being said, in some cases the solver creates a file in the hard drive to partially store tree nodes of the branch-and-cut when it is too big to fit in the memory of the machine. However, this did not happen for simulations in Case 1 because of the performance optimizations. Actually, IBM's CPLEX® solver was executed at the middle of adding tracks for mission 6, mission 8 and mission 10, and at the end of each simulation. This made each execution of the solver to use less memory than if this optimization was not implemented.

3.2 Case 2: 2-month full DSN scenario

The input data for this case corresponds to a real full DSN scenario for two months. There are 34 different missions requesting DSN usage within that period. There are many more visibility input files than Case 1 because now each mission has at least

one unique visibility file. The DSN architecture evolution file provided as input is the current DSN architecture without any change during all period of interest. The input has 211 operational segments that generate 3,620 tracks. As in the previous sections, we first analyze the schedule results and then report memory and time performance metrics.

3.2.1 Analysis of the results

From the 3,620 tracks requested, 2,457 tracks are scheduled. This corresponds to the 68% of the tracks requested. On the other hand, 7,203 of 12,928 hours requested are finally scheduled. This corresponds to the 58% of the hours requested. There is a notable difference between both percentages because the MILP problem generally prioritizes scheduling as many tracks as possible, regardless of their duration. Furthermore, scheduling several short tracks can use the same amount and time of resources as scheduling one large track. Therefore, we conclude from these results that the average length of scheduled tracks is smaller than the average length of the tracks requested.

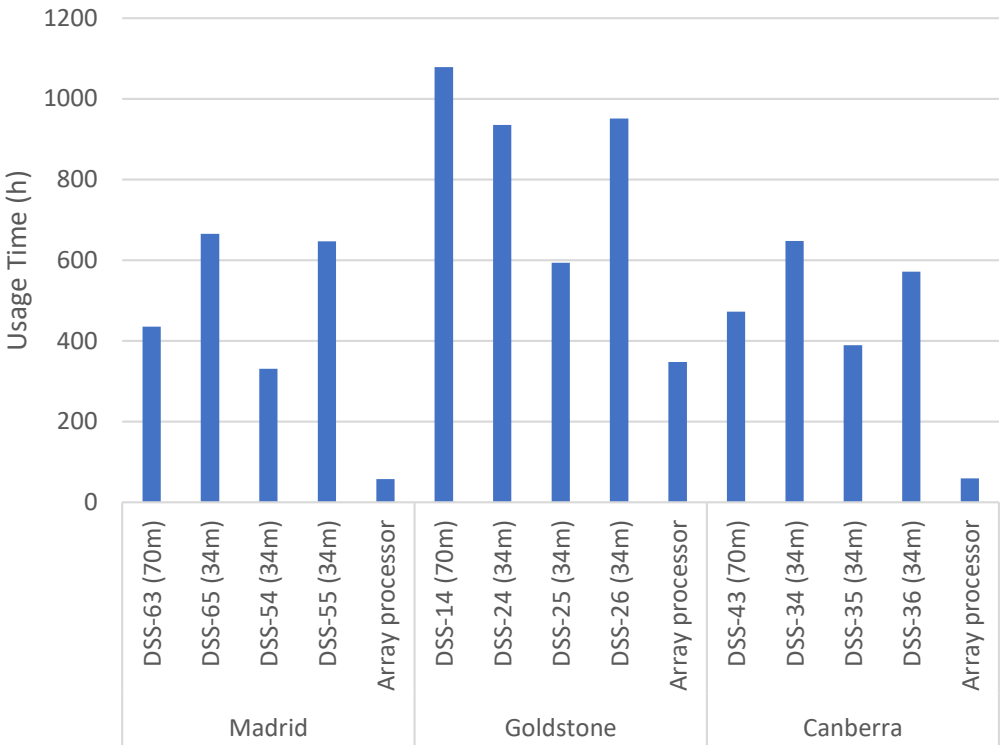

Figure 3-3: Usage time of DSN resources for 2 months (Case 2)

Figure 3-3 plots the total amount of hours that each DSN resource is used for the 2 months. DSN resources considered are the Deep Space Stations (antennas) and the array processor of each ground station. The figure indicates that resources in GDSCC –Goldstone Deep Space Communication Complex– generally receive more workload than resources in other sites. Since the characteristics of the antennas are

very similar from site to site, probably this happens because Goldstone complex have longer visibility periods with space missions than other complexes because of their location.

Finally, Figure 3-3 represents a real example of how the toolkit developed in this research will be used. Indeed, loading analysis shown in these types figures could be used to strategically plan future additional capabilities of the Deep Space Network.

3.2.2 Computational performance

The overall computational performance of this case using the current status of the LTST is not as good as expected. The parameters of the software have been configured to do 33 intermediate executions of the IBM's CPLEX® solver following the optimization explained in Section 2.4.4. These executions last between 1 and 11 seconds, in increasing order as more executions are performed.

The final solver execution incurs in significantly more time than all others. In fact, the solver did not finish completely and it was instead terminated after one hour of runtime. That being said, note at at this point the optimality level was already at 86% and therefore it becomes increasingly difficult to find better solutions in a very large space of possibilities. It is coherent to think this because, given the input data of this case, with a batch of tracks from different missions overlapping within a period of only 2 months under consideration, the result that 68% of tracks are scheduled was expected and is considered correct.

Memory management followed a similar tendency as time performance. Intermediate executions of the solver do not use to much computer memory, because they finish soon. In contrast, the last execution of the whole problem uses more memory as time passes. For instance after 8 minutes the solver created a file to store tree nodes in the hard drive. After an hour of execution, the compressed size of the file was more than 6 gigabytes. Improvement of the computational performance –both in time and memory– for this case should be conducted in the future, as explained in Section 4.3.

THIS PAGE INTENTIONALLY LEFT BLANK

4. Conclusions

4.1 Thesis Summary

The Deep Space Network is very expensive to build, maintain and operate. To address this issue, additional system capabilities are planned strategically, years in advance, by forecasting which missions will utilize the system in the coming decades. Therefore, loading analyses are conducted assuming different scenarios, each one simulating DSN operations for several years.

This thesis provides an automated long-term scheduling toolkit that can mimic real DSN operations. Several factors are modeled and accounted for in this process: spacecraft visibility constraints, evolution of the DSN architecture, characteristics of each antenna, as well as link and other operational constraints.

Chapter 1 introduced the main concepts of this research. First, the context and motivation for this research were presented. Then, the state-of-the-art in DSN scheduling was reviewed. Next, some background on DSN and DSN scheduling was provided. Afterwards, the general problem statement was defined. Methods to solve "The DSN Scheduling Problem" were analyzed in the literature review, and determined that the best option was to formulate an optimization problem using Mixed Integer Linear Programming. Finally, the thesis statement was presented.

Chapter 2 explained in detail how the LTST formulates the MILP problem. It first presented some definitions necessary for the following sections. Then, it defined the objective function of the optimization problem. The next section detailed the steps involved in formulating the MILP problem and the last section in this chapter mathematically presented the MILP problem.

Chapter 3 presented and analyzed the results of several case inputs, using real NASA missions and DSN inputs. Case 1 replicated a 22.5-year mission several times, imposing hard conditions to the problem. Case 2 simulated a 2-month full DSN scenario. It also discussed computational performance of the toolkit.

Finally, this chapter summarizes the work conducted in this thesis, identifies its main findings and contributions, and highlights opportunities for future work.

4.2 Thesis Contributions

The main contributions of this research are summarized in the following list:

1. **Review** the state-of-the-art in DSN scheduling software. Section 1.2.1 reviewed the state-of-the art in DSN scheduling. It specially focused on the algorithms and techniques formulated before this research. Then, each software or work

presented in that section was discussed as to why it was not enough for the purpose of this thesis, and why the research conducted in this thesis was necessary.

2. **Identify** the elements and factors involved in long-term DSN scheduling, divided into two parts. The first part presented the main elements of the Deep Space Network. The second part identified the elements and factors involved in DSN scheduling.

3. **Characterize** the elements and factors involved in long-term DSN scheduling. In the same background Section 1.3, as well as throughout Chapter 2, everything involved in long-term DSN scheduling was characterized. The characteristics of the elements were enumerated and considered whether they should be taken into account for this research or not. The basic elements considered were: deep space missions, operational segments, tracks, links and visibility.

4. **Analyze** and compare methods for solving the problem. Section 1.5 presented several methods and algorithms to solve the assignment problem and the covering problem involved in "The DSN Scheduling Problem". These were analyzed and compared, and Mixed Integer Linear Programming (MILP) was the option finally selected.

5. **Formulate** the mixed integer linear programming problem. Chapter 2 extensively explains the algorithm to formulate the MILP problem. Decisions on this generation algorithm are justified. This and the following contributions listed here are the most important in this research.

6. **Develop** an automated long-term scheduling toolkit for the DSN (LTST). A complete software has been developed using Python programming language and the Python API of the IBM's CPLEX® solver is used to solve the MILP problem. The implementation of this toolkit is modular, splitting the source code in several files.

7. **Optimize** the resolution of the MILP problem. Due to the complexity of the problem at hand, some simulations were very slow to execute. Thus, some important optimizations were implemented in the software. Section 2.4.4 explained the performance optimizations implemented.

8. **Present** the results and performance of two test cases including time performance and memory management.

The present research conducted by the author at Jet Propulsion Laboratory concludes with the implementation of an automated long-term scheduling toolkit for the Deep Space Network, and the writing, submission and defense of this thesis.

4.3 Future Work

There exists several areas for future work within this research. These areas can be grouped in two categories: (1) Future work to add or improve features of the scheduling toolkit; and (2) future work for improving time performance, memory managing, and scalability of the resolution of the MILP problem. Within the first category, one

feature to be implemented is considering the Sun-Earth-Probe (SEP) angle as a factor to take into account while scheduling passes.[4] When this angle (in absolute value) is small, a connection between Earth and the spacecraft cannot be established because of the noise added by the Sun's radiation. Note that this value is already provided in the visibility input files. Therefore, the visibility should be shortened so that does not include small SEP angles. However, the availability of the connection depends on the frequency band. For instance, while Ka-band communication is assumed to be unavailable for SEP angles of less than 1°, X-band communication is assumed unavailable for SEP angles of less than 3° [28]. Therefore, the visibility due to the SEP angle will depend on the frequency band of the track.

Another feature to improve would be to take into account the duration of the operational segment when creating the "track options" weights. For example, if there is an operational segment with just one track, and the length of the track is almost the same as the length of the operational segment, Equation (2.3) should give the track a very high weight. This is due to the fact that not scheduling that track would be very bad as it is the only track of an entire operational segment.

An advanced feature to be added would be the formulation of complex "track options". For example, instead of a track having a fixed length and data rate, a "track option" could have a specific data rate according to the antennas used and the characteristics of the spacecraft. Therefore, if different "track options" for the same track have different data rates, the length of the track would depend on which option is finally chosen. Another advanced "track option" would be allowing split tracks. In other words, when a track is too long, the LTST could automatically split the track into two close tracks with equal characteristics, that end up transmitting the same total amount of data as the original requested track.

The second category defined for future work –improving time performance, memory managing, and scalability of the resolution of the MILP problem– is the major challenge. This thesis mainly focused on the generation process of the problem, although it included some performance optimizations in Section 2.4.4. However, these optimizations were added at the end of this research because they were necessary to run all the tests and obtain results. Therefore, more tuning of the problem parameters and the solver's configuration parameters should be carried out. For example, improving the time instances when the partial problem is solved would be key for performance improvement. Now, the solver is defined to run when certain amount of MILP variables and constraints are achieved. However, small problems should still need to have intermediate executions of the solver, even if they have less variables and constraints than bigger problems that do not need intermediate executions of the solver. This can happen because some constraints can restrict the MILP problem more than other constraints. Big amounts of tracks of different missions within the same time region generate a MILP problem difficult to solve. Therefore, while these tracks are added, some execution of the solver will be helpful for the overall performance of the toolkit.

[4] Angle between the straight lines from Earth to the Sun and from Earth to the probe (mission).

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] Leslie Deutsch. NASA's Deep Space Network: big antennas with a big job, 2012. URL https://trs.jpl.nasa.gov/bitstream/handle/2014/42668/12-1936_A1b.pdf.

[2] Glen Nagle and Korinne McDonnell. Canberra Deep Space Communication Complex - NASA's Deep Space Network, 2017. URL https://www.cdscc.nasa.gov/Pages/antennas.html. Accessed on 2019-06-10.

[3] David Doody. Basics of Space Flight. Chapter 18: Deep Space Network, 2017. URL https://solarsystem.nasa.gov/basics/chapter18-1/. Accessed on 2019-05-31.

[4] James Zumberge, Leslie Deutsch, and Steve Townes. Deep Space Communications, 2017. URL https://scienceandtechnology.jpl.nasa.gov/research/research-topics-list/communications-computing-software/deep-space-communications. Accessed on 2019-05-31.

[5] Marc Sanchez Net. *Support of Latency-sensitive Space Exploration Applications in Future Space Communication Systems*. PhD thesis, Massachusetts Institute of Technology Department of Aeronautics and Astronautics, Cambridge, Massachusetts, jun 2017.

[6] Mark Johnston and Jigna Lad. Integrated Planning and Scheduling for NASA's Deep Space Network – from Forecasting to Real-time. In *15th International Conference on Space Operations*, Reston, Virginia, may 2018. American Institute of Aeronautics and Astronautics. ISBN 978-1-62410-562-3. doi: 10.2514/6.2018-2728.

[7] Butch Carruth, Mark Johnston, Adam Coffman, Mike Wallace, Belinda Arroyo, and Shan Malhotra. A Collaborative Scheduling Environment for NASA's Deep Space Network. 2012. doi: 10.2514/6.2010-2284.

[8] Mark Johnston. Automating Mid- and Long-Range Scheduling for NASA's Deep Space Network. In *SpaceOps 2012 Conference*, Reston, Virigina, jun 2012. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2012-1296235.

[9] Mark D. Johnston, Daniel Tran, Belinda Arroyo, and Chris Page. Request-Driven Scheduling for NASA's Deep Space Network. In *6th International Workshop on Planning and Scheduling for Space (IWPSS-09)*, 2009.

[10] Kar-Ming Cheung, C.H. Lee, W.B. Gearhart, T. Vo, and S. Sindi. Link-capability driven network planning and operation. In *Proceedings, IEEE Aerospace Conference*, volume 7, pages 7–3281–7–3285. IEEE. ISBN 0-7803-7231-X. doi: 10.1109/AERO.2002.1035304.

[11] Yijiang Chen, Douglas S. Abraham, David P. Heckman, Andrew Kwok, Bruce E. MacNeal, Kristy Tran, and Janet P. Wu. Architectural and operational considerations emerging from hybrid RF-optical network loading simulations. In Hamid Hemmati and Don M. Boroson, editors, *Free-Space Laser Communication and Atmospheric Propagation XXVIII*, volume 9739, page 97390P, 2016. doi: 10.1117/12.2213594.

[12] Colin E. Bell. Scheduling deep-space network data transmissions: a Lagrangian relaxation approach. volume 1963, pages 330–340. International Society for Optics and Photonics, mar 1993. doi: 10.1117/12.141750.

[13] E. Kan, J. Rosas, and Q. Vu. Operations mission planner - 26m user guide modified 1.0. Technical report, Jet Propulsion Laboratory, 1996.

[14] S.A. Chien, R.W. Hill, A. Govindjee, X. Wang, T. Estlin, M.A. Griesel, R. Lam, and K.V. Fayyad. A hierarchical architecture for resource allocation, plan execution, and revision for operation of a network of communications antennas. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 3340–3347. IEEE, 1997. ISBN 0-7803-3612-7. doi: 10.1109/ROBOT.1997.606798.

[15] Mark D. Johnston. Multi-Objective Scheduling for NASA's Future Deep Space Network Array. In *5th International Workshop on Planning and Scheduling for Space (IWPSS-06)*, 2006.

[16] Thuy Mai. Deep Space Network, 2018. URL `https://www.nasa.gov/directorates/heo/scan/services/networks/dsn`. Accessed on 2019-06-09.

[17] Andrew Kwok. 201, Rev. B: Frequency and Channel Assignments. Technical report, Jet Propulsion Laboratory, 2009. URL `https://deepspace.jpl.nasa.gov/dsndocs/810-005/201/201B.pdf`.

[18] N. Renzetti. DSN Functions and Facilities. Technical report, Jet Propulsion Laboratory, 1975. URL `https://ipnpr.jpl.nasa.gov/progress_report2/42-28/28A.PDF`.

[19] Kevin Hussey, Michael Rodrigues, Shannon McConnell, Michael Levesque, Doug Ellison, Stephen Hurley, Erik Boettcher, Davit Stepanyan, Kristine Nguyen, Daren Lee, Bach Bui, Jim McClure, Dan Goods, Brian Kumanchik, Christian Lopez, Justin Moore, and Andrea Boeck. DSN Now, 2019. URL `https://eyes.nasa.gov/dsn/dsn.html`. Accessed on 2019-06-09.

[20] Bruce E. MacNeal, Douglas S. Abraham, Rolf C. Hastrup, Janet P. Wu, Richard J. Machuzak, David P. Heckman, Robert J. Cesarone, Raffi P. Tikidjian, and Kristy Tran. Mission set analysis tool for assessing future demands on NASA's Deep Space Network. In *2009 IEEE Aerospace conference*, pages 1–11. IEEE, mar 2009. ISBN 978-1-4244-2621-8. doi: 10.1109/AERO.2009.4839377.

[21] Michael L. Pinedo. Deterministic Models: Preliminaries. In *Scheduling*, pages 13–32. Springer International Publishing, Cham, 2016. doi: 10.1007/978-3-319-26580-3_2.

[22] Harold W. Kuhn. The Hungarian Method for the Assignment Problem. In *50 Years of Integer Programming 1958-2008*, pages 29–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. doi: 10.1007/978-3-540-68279-0_2.

[23] Dimitri P. Bertsekas and David A. Castanon. The auction algorithm for the transportation problem. *Annals of Operations Research*, 20(1):67–96, dec 1989. ISSN 0254-5330. doi: 10.1007/BF02216923.

[24] Gerard Sierksma and Yori Zwols. *Linear and integer optimization : theory and practice*. CRC Press, 2015. ISBN 9781498710169.

[25] Telecom ABC. Equivalent Isotropically Radiated Power, 2008. URL `http://www.telecomabc.com/e/eirp.html`. Accessed on 2019-06-12.

[26] Wikipedia. Antenna gain-to-noise-temperature, 2017. URL `https://en.wikipedia.org/w/index.php?title=Antenna_gain-to-noise-temperature&oldid=759398933`. Accessed on 2019-06-12.

[27] Consultative Committee for Space Data Systems. TM Synchronization and Channel Coding - Summary of Concept and Rationale. Technical report, 2012. URL `https://public.ccsds.org/Pubs/130x1g2.pdf`.

[28] Yanping Guo, James Mcadams, Martin Ozimek, and Wen-Jong Shyong. Solar Probe Plus Mission Design Overview and Mission Profile. In *International Symposium on Space Flight Dynamics*, 2014.